UNIVERSITY OF CALIFORNIA,
IRVINE

Leveraging Community Structure and Behavior for Smart Infrastructure

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Praveen Venkateswaran

Dissertation Committee:
Professor Nalini Venkatasubramanian, Chair
Professor Nikil Dutt
Professor Tony Givargis
Professor Sharad Mehrotra

2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

My journey has been largely possible due to people taking a chance on me at various points of time, as well as the collaborators and well-wishers who have helped me along this path and provided their friendship. I would like to thank,

My advisor, Nalini Venkatasubramanian, for the different research opportunities, constant support and career guidance, and most importantly, for helping me grow into an independent researcher.

My committee, Nikil Dutt, Tony Givargis, and Sharad Mehrotra, for their feedback and suggestions during my examinations that helped shape this thesis.

My many collaborators. Cheng-Hsin Hsu and Chia-Ying Hsieh for their invaluable help in building the REAM framework, and for helping me improve my systems building skills. Mahima Agumbe Suresh, for her tireless efforts and energy despite having to collaborate remotely, resulting in our work on sensor deployment. Qing Han, for teaching me the ropes of the PhD and the AquaSCALE project, and for constantly helping and looking out for me. Kyle Benson, for providing me the opportunity and expertise needed to enhance and complete the REAM framework.

My friends at UCI. My fellow students in the DSM and ISG groups, for their support and encouragement during the good and the tough times. Gift Sinthong and Siddharth Gupta, for always having my back, and providing unique perspectives and wonderful conversations that helped me grow.

My internship mentors, Minyoung Kim and Carolyn Talcott at SRI, Sorin Iftimie and Madhumita Dange at Microsoft, and Vinod Muthusamy, Vatche Isahagian and Evelyn Duesterwald at IBM Research. I am truly grateful for the wonderful opportunities I received to intern at these different places, and work with incredibly smart and kind people who were always willing to share their knowledge and mentor me.

My undergrad mentors, Maitreya Natu and Vaishali Sadaphal, for the time and energy spent in teaching me how to do research, for changing my outlook on life, and for your kindness and friendship. Aruna Malapati, for going above and beyond in providing me opportunities to get into research, and for being someone I could always count on for support. Without them, this journey would never have begun, and I can only hope to emulate them.

My parents, for always supporting my education and their sacrifices to ensure that I could pursue my studies. I am grateful for their lifelong advice and for always believing in me.

My partner, Saumya Gupta, for being my biggest supporter, for your immense patience during the tough times, and for always encouraging me to set lofty goals and achieve them. You constantly inspire me to become a better person and to enjoy life to its fullest. Thank you for making this journey brighter and more fulfilling, and most importantly, thank you for being my best friend.

My PhD research was supported in part by the National Science Foundation (NSF) awards CNS-1528995 and CNS-1952247, and the Donald Bren School of Information and Computer Sciences (ICS) at the University of California, Irvine.

VITA

Praveen Venkateswaran

EDUCATION

Doctor of Philosophy in Computer Science — **2021**
University of California, Irvine — *Irvine, California*

Master of Science in Mathematics — **2016**
Birla Institute of Technology and Science, Pilani — *Hyderabad, India*

Bachelor of Science in Computer Science — **2016**
Birla Institute of Technology and Science, Pilani — *Hyderabad, India*

RESEARCH EXPERIENCE

Graduate Student Researcher — **2016–2021**
University of California, Irvine — *Irvine, California*

PROFESSIONAL EXPERIENCE

Data and Applied Scientist Intern — **Summer 2021**
Microsoft — *Redmond, Washington*

Research Intern — **Summer 2020**
IBM Research — *Cambridge, Massachusetts*

Data and Applied Scientist Intern — **Summer 2019**
Microsoft — *Redmond, Washington*

Research Intern — **Summer 2018**
Stanford Research International — *Palo Alto, California*

TEACHING EXPERIENCE

Teaching Assistant — **2016–2021**
University of California, Irvine — *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

REAM: A Framework for Resource Efficient Adaptive Monitoring of Community Spaces **Sep 2021**
Pervasive and Mobile Computing

REFEREED CONFERENCE PUBLICATIONS

Robust and Generalizable Predictive Models for Business Processes **Sep 2021**
International Conference on Business Process Management (BPM)

Environment Agnostic Invariant Risk Minimization for Classification of Sequential Datasets **Aug 2021**
ACM Conference on Knowledge Discovery and Data Mining (KDD)

REAM: Resource Efficient Adaptive Monitoring of Community Spaces at the Edge Using Reinforcement Learning **Sep 2020**
IEEE International Conference on Smart Computing (SMARTCOMP) [*Best Paper*]

Augmenting In-situ with Mobile Sensing for Adaptive Monitoring of Water Distribution Networks **Apr 2019**
ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)

Impact Driven Sensor Placement for Leak Detection in Community Water Networks **Apr 2018**
ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)

ABSTRACT OF THE DISSERTATION

Leveraging Community Structure and Behavior for Smart Infrastructure

By

Praveen Venkateswaran

Doctor of Philosophy in Computer Science

University of California, Irvine, 2021

Professor Nalini Venkatasubramanian, Chair

Modern community infrastructures are increasingly instrumented with Internet of Things (IoT) sensors and actuators, which enable many essential infrastructure monitoring applications. These applications are now ubiquitous across domains such as smart transportation, power grid, ambient environment sensing, smart buildings, among others, and provide important real-time information about the infrastructure and enable the accurate detection of critical events. A typical infrastructure monitoring framework involves data collection from distributed deployments of sensors, transmission over communication networks, and analysis using analytical models at edge and cloud servers to generate meaningful and actionable information.

However, communities exhibit heterogeneity in their structure and behavioral patterns. Structural heterogeneity can manifest through differences in topography, infrastructure scale and layout, community demographics, and available monitoring resources, while behavioral diversity can occur due to differences in weather phenomena, spatio-temporal patterns like vehicular movement, and other infrastructure activities. Current monitoring approaches are limited by this heterogeneity, and only work for specific communities and applications.

In this thesis, we propose solutions to leverage this heterogeneity to build effective, efficient, and adaptive infrastructure monitoring applications that can be deployed and shared

across communities. Our proposed techniques leverage the unique structural and behavioral characteristics of communities, while also balancing monitoring requirements of applications with infrastructure resource availability. We explore our approach within the context of several real-world infrastructure monitoring applications and address three research problems across sensor deployment, operation of monitoring applications, and the generalization of monitoring solutions across communities.

First, we propose an impact-driven approach to IoT sensor placement that leverages community characteristics to determine vulnerable regions and measures the potential impact of events which is used to prioritize deployment locations. Second, we design an operational monitoring framework that handles heterogeneity in devices, communication networks, and analytical models and develop an adaptive decision making approach to determine the optimal choices for monitoring while balancing performance, resource consumption and current community conditions. Finally, we present an approach that enables training robust infrastructure monitoring models from multiple data sources in a distributed and bias-agnostic manner, that can then generalize or be reused across communities without a loss in performance. Together, the proposed techniques provide a comprehensive approach for infrastructure monitoring that can exploit and adapt to structural and behavioral characteristics of communities. We validate our approach using prototype implementations on several real-world infrastructure testbeds.

Chapter 1

Introduction

In this chapter, we introduce the use of the Internet of Things (IoT) in smart communities and present a discussion of the major characteristics of IoT-driven community infrastructure monitoring frameworks. Distributed deployments of IoT devices in community infrastructure, ranging from buildings, roads, power grids, etc., can provide near real-time monitoring capabilities and timely detection of events within the infrastructure. Enabling community infrastructure monitoring requires the collection and delivery of data from deployments of heterogeneous devices over a diverse set of communication networks, followed by their analysis using complex analytical models to obtain meaningful and actionable information. We present key challenges across these different components within an IoT-driven infrastructure monitoring framework. We further illustrate them through a driving use-case of drinking and storm water infrastructure monitoring, and describe the efforts in this thesis towards addressing these challenges. In particular, we strive to leverage the diverse characteristics of community structure and behavior to develop solutions for the deployment, operation, and generalization of infrastructure monitoring frameworks.

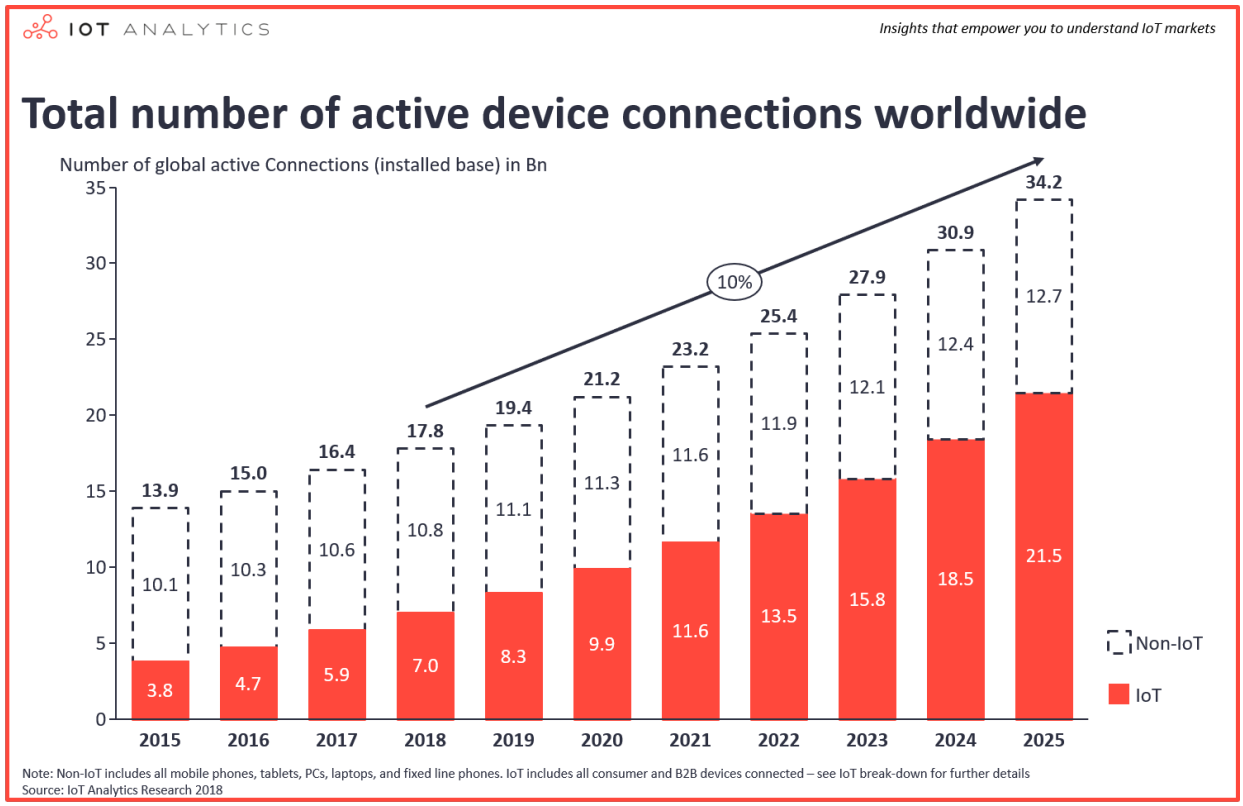


Figure 1.1: Growth of IoT devices in communities as a proportion of total number of global connected devices

1.1 Internet of Things for Community Infrastructure

The Internet of Things (IoT) refers to the billions of physical devices and everyday objects around the world that are now connected to the internet while collecting and sharing data and information. With the addition of sensing, computation and networking capabilities, these objects are capable of communicating with each other and other cloud and edge services across the Internet. The past decade has seen the arrival of cheap components and sensors (e.g., Raspberry Pi in 2012) and the ubiquity of wireless networks across the world. This, coupled with the increased accessibility to compute and analytics resources on public or private cloud and edge servers, has resulted in a significant growth of IoT device presence in communities (Figure 1.1).

The resulting economies of scale from this growth has improved the affordability of IoT de-

ployments and has made IoT-driven monitoring available to even lower-income and resource-poor communities and populations. A recent market report by IoT Analytics [81] shown in Figure 1.1, projects an 85% growth in the number of IoT devices in the next 4 years, and also projects that the number of IoT devices around the world would outnumber the number of non-IoT devices by nearly 8.8 billion devices (69%) by 2025. This growth in IoT is also reflected in the market size (Figure 1.2), where the global IoT market was $457 billion in 2020, a growth of nearly 200% from 2016.

The Internet of Things is now seeing significant use by community agencies and stakeholders to implement their vision of smart communities and smart cities (Figure 1.2). IoT devices and sensors are being deployed and utilized by different monitoring applications to enhance the quality and performance of services, reduce costs and resource consumption, and to ensure real-time event detection and incident response, among other uses, by collecting and analyzing the data streams. There are many monitoring applications across different community infrastructure that are currently being used, such as personal healthcare monitoring, smart street lighting, occupancy and intruder detection in buildings, traffic monitoring, etc. IoT is also seeing increased use for monitoring large-scale community infrastructure such as the power grid and water distribution networks. Each of these applications exhibit diversity and heterogeneity in their monitoring scope (e.g., individuals vs. buildings), objectives, as well as their requirements for sensing, communication, and analytics. Additionally, communities themselves exhibit heterogeneity that can impact the functioning and performance of these monitoring applications. For example, weather patterns like precipitation and temperature, structural factors like infrastructure scale and coverage, can differ both within and across communities, thereby requiring applications to adapt to these different conditions.

This inherent heterogeneity in communities and infrastructures, coupled with the diversity in monitoring application requirements presents a multitude of unique challenges. In this thesis, we address these challenges by leveraging the heterogeneity to support and develop

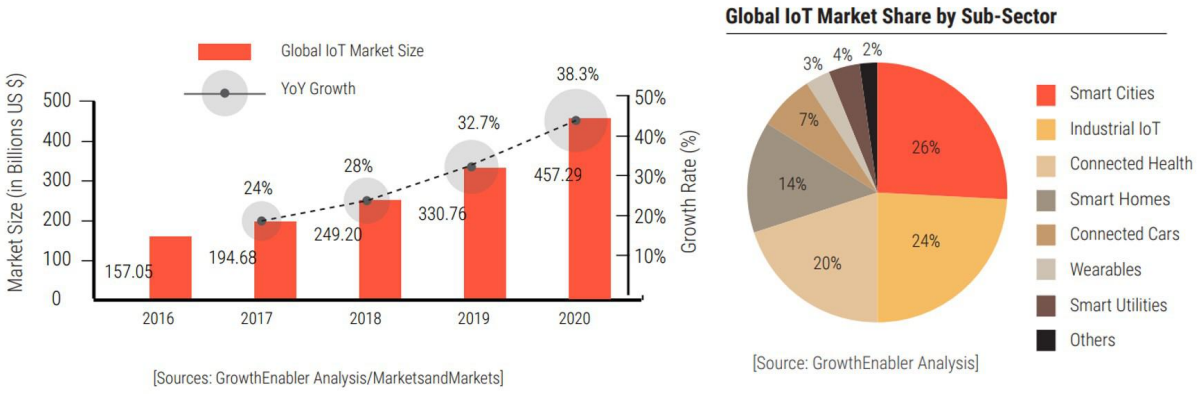


Figure 1.2: Growth of global IoT market spend and importance of smart communities

effective, efficient, and adaptive infrastructure monitoring solutions.

1.2 Community Infrastructure Monitoring

The functioning of communities revolves around their built infrastructure. Given the importance of some community infrastructure like the power grid, water distribution networks, and traffic systems, among others, the ability to deploy monitoring applications that provide information about the infrastructure is critical to maintain their health and smooth functioning. This emphasis on infrastructure monitoring using IoT can be seen in Figure 1.3, which shows the most popular uses of IoT for smart communities. A vast majority of use-cases involve monitoring of different kinds of community infrastructure, thereby also showing the immense potential of leveraging IoT for building smart infrastructure monitoring solutions.

Effective infrastructure monitoring allows community agencies and stakeholders to obtain comprehensive continuous information about the state of the entire system, thereby allowing them to accurately detect events in a timely manner, or even proactively predict them well in advance. These events can range in their effect, complexity, importance, and their impact on the community infrastructure. For instance, detecting the presence of a person in

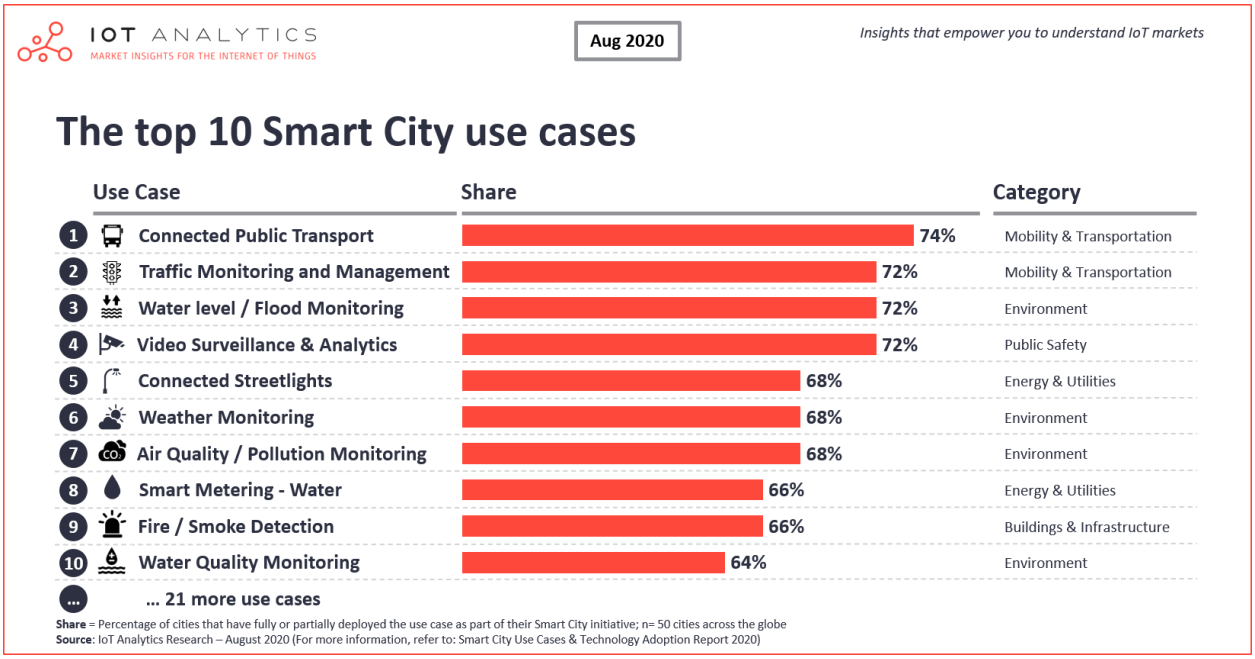


Figure 1.3: Importance of Infrastructure Monitoring Applications in Smart Communities

a building is significantly different compared to detecting air pollution or water contamination. The ability, or the inability, to accurately detect events in a timely manner can impact the effectiveness of infrastructure monitoring solutions. Missing infrastructure events or even a delay in detecting them can cause a significant adverse impact on the community. For instance, in Flint, Michigan, a failure to detect the presence of lead in drinking water resulted in a huge crisis that impacted thousands of residents for more than half a decade [25]. Similarly, the lack of sufficient monitoring solutions led to the delayed detection of transmission line failure at a power substation in North Gila, Arizona [148]. This incident resulted in widespread power blackouts across Arizona, Orange County, and San Diego leading to loss of power for over 2 million homes. This caused cascading issues in other industries like grocery stores, gas stations, and even hospitals, thereby creating significant adverse impacts on numerous communities. On the flip side, developing a comprehensive end-to-end monitoring framework can improve the robustness of community infrastructure to different events, and empower agencies and stakeholders to provide effective and efficient service to the citizens of the community. For example, in Alabama, following Hurricane Ida in September 2021,

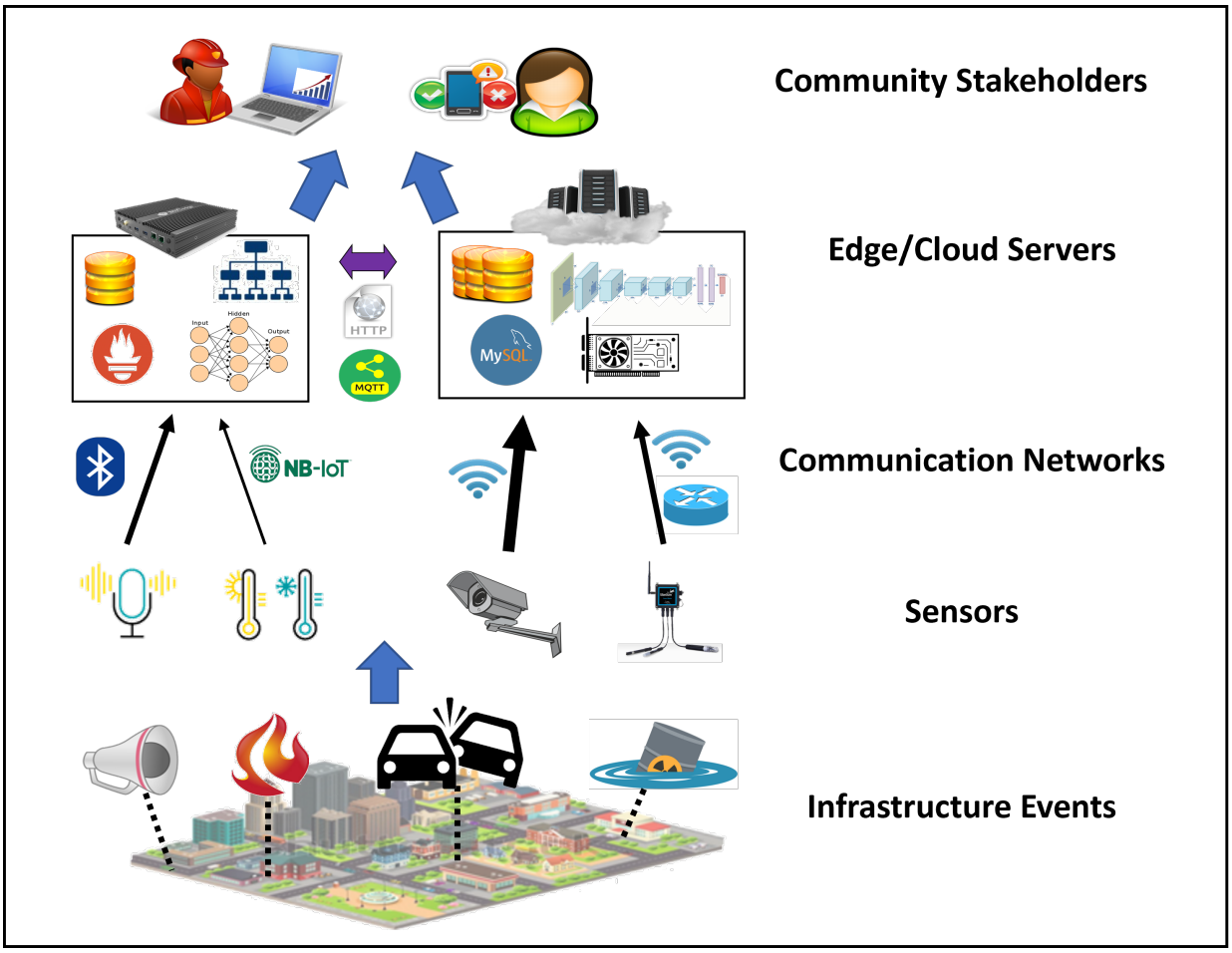


Figure 1.4: Community Infrastructure Monitoring Workflow

the use of smart grid monitoring technology enabled rapid fault detection across the entire grid. It automatically identified more than 500 faults quickly after the event, which helped direct agency crews to outages and reduce by 137 miles the distance that they needed to patrol. This further helped save 2.5 million customer minutes of interruption and reduced the duration of outages by 1.3 million minutes [29].

Irrespective of the type of community infrastructure, every comprehensive monitoring framework requires several different layers. While the specifics of these layers may differ across different infrastructure or monitoring applications, they together form an end-to-end monitoring workflow that is essential for an effective monitoring framework. Figure 1.4 depicts these different layers and the relationship between them. At the lowest layer, we have the

community infrastructure itself, where events of different kinds (e.g., fire, traffic accidents, person detection) in the community are detected by a distributed network of deployed sensors. The data streams from these sensors are transmitted over a variety of communication network links, which can be both wired or wireless (e.g., WiFi, bluetooth) and have varying connectivity strengths and available bandwidth. A set of edge and/or cloud servers receive the data streams, where each server has specific resources allocated towards data storage, computation capabilities, and available analytical models (e.g., image classifiers). The data streams are passed as input to different analytical models to obtain actionable information that is subsequently accessed by different stakeholders or users in the community, to provide them with comprehensive monitoring information about the community infrastructure.

While the above figure depicts the common layers of an infrastructure monitoring framework, the specific features within each layer can be quite different depending on the type of infrastructure as well as the monitoring requirements. For instance, the type of sensors, network links and analytical models needed for contamination detection in water networks will be different from those needed for occupancy monitoring in a smart building. Moreover, an additional challenge is that this variance is also exhibited across communities and community spaces as well. Communities can differ in terms of their (i) structure – topography, types of infrastructure, population demographics, access to capital to instrument IoT sensors, compute resource availability, etc. and (ii) behavior – weather and climate patterns, pedestrian and vehicular movement, event correlations, unique events like festivals, which influence the spatio-temporal patterns that generate events. This diversity and heterogeneity in community structure and behavior present several challenges that need to be addressed and leveraged if we are to build a comprehensive infrastructure monitoring framework across these different layers.

1.3 Key Challenges

Today's modern communities have the ability to deploy large-scale IoT sensors and distributed storage and compute servers at the cloud and edge that can leverage sensing, networking, compute, and community characteristics to enable real-time adaptive monitoring. Such real-time monitoring frameworks have to detect events in the community in a timely and proactive manner, and support the seamless recovery of community infrastructure when adverse events occur. An effective monitoring framework can provide real-time information and event detection, reduce resource costs, minimize disruption to the community due to adverse events, while also adapting to changing community conditions. However, there does not yet exist a comprehensive monitoring framework that achieves all these different objectives simultaneously. The development and implementation of such an adaptive infrastructure monitoring framework has been hindered by the following challenges.

Diversity in Community Structure and Behavior

Numerous communities and community spaces can be found across the globe and their number is ever-increasing. These communities exhibit tremendous diversity along many axes. These can include structural and geosocial differences like population scale and demography, topographical differences, level of urbanization and infrastructure development, economic health and ease of access to monitoring resources, among others. For example, the monitoring requirements, scale of operations, and level of monetary capital of a large city like Los Angeles would be very different to those of a small, relatively unpopulated town in Idaho. Moreover, even among communities with similar structure, the inherent behavior and spatio-temporal characteristics can often vary. These behavioral patterns can include weather phenomena, vehicular and pedestrian movement, temporal occurrence and correlations of events, among others. For instance, an industrial region would exhibit movement

and activity across all hours of the day, while a residential area would predominantly show diurnal patterns. This challenge of diversity in community structure and behavior has resulted in existing infrastructure monitoring efforts to focus on specific communities and applications, and there has been no significant effort towards building cross-community monitoring frameworks. While this diversity presents a challenge, it also provides a rich opportunity for collecting diverse information of data and events which, if leveraged, can result in monitoring frameworks that can adapt to numerous events and scenarios observed across communities.

Heterogeneity in Devices and Analytical Models

The popularity of the Internet of Things (IoT) has resulted in an explosion in the quantity and heterogeneity of devices available in the market today (Figure 1.2). These can range from simple acoustic sensors such as those to measure temperature, PM 2.5, humidity, sound, etc., to more sophisticated devices like cameras and smartphones. There also exist specialized sensors for community infrastructure like SCADA systems for water networks and smart grid sensors. This also results in heterogeneity in the types of data streams from these sensors (e.g., time-series, images, geospatial), where these streams can be used by a variety of monitoring applications depending on their objectives. Heterogeneity can exist even among sensors of the same type. They can vary in cost, size, sensing modality, effective range, networking capabilities (some can use multiple types of networks like bluetooth, wifi, etc.) and additional features like ruggedness, data generation patterns (e.g., streaming, periodic), among others. Heterogeneity can also be observed in the analytical models used to analyze the data streams from these devices which can range from simple rule-based heuristics to complex machine learning models. These models can vary in terms of their input features, performance, architecture, and even resource requirements like power, compute, and data storage. The performance of sensors, networks, and analytical models are closely tied to a given community environment, and hence understanding the effectiveness of each option,

and adapting the choice of sensing, communication, and analytics in real-time to changing conditions is essential for a good infrastructure monitoring framework.

Application Diversity and Real-time Adaptation

Existing efforts for community infrastructure monitoring typically focus on one application. However, as sensing and computing capabilities develop, communities increasingly wish to simultaneously deploy multiple monitoring applications like pollution monitoring, smart transportation, surveillance for safety, and smart buildings, to name a few. These applications are diverse in their objectives, sensing and analytics requirements, as well as their importance to each community. Rather than running these diverse applications independently, which can result in redundancy of resource consumption and data collection, a smart infrastructure monitoring framework can identify and leverage any commonality between them. This can be done for (a) data sharing – where the same sensors can be used by multiple applications (e.g.) using traffic cameras for both accident detection and pedestrian surveillance, (b) analytical models – where strong correlations between events, multiple infrastructure features or measurements can be leveraged for better performance (e.g.) correlation between air pollution and high vehicular traffic, (c) prioritization – critical applications like fire detection should always have priority for resources over more routine applications like rainfall monitoring. It is also important for an effective monitoring framework to be able to adapt to changes in the community in real-time by intelligently identifying and switching between different sensing, communication, and analytics options, while balancing monitoring performance with the judicious use of resources.

Distributed Training and Reusability of Monitoring Models

Since infrastructures in different communities can observe a diverse range of events, being able to train monitoring models on data shared by these different communities can improve the robustness of the models to more events. Additionally, analytical models that can be deployed or reused in multiple communities without a loss in performance provide significant benefits. This generalizability would allow communities to save on the cost, time, and effort needed for instrumentation and data collection, where instead they could just obtain models from other communities. Moreover, community efforts towards infrastructure monitoring can often be limited by access to resources such as the amount of available capital, instrumentation, available compute resources and data that are needed to train effective analytical models. This results in communities with access to different levels of resources having starkly different monitoring capabilities both in terms of the quality and extent of monitoring, demonstrating the benefits of reusable or generalizable models.

However, centralized training of models requires transmission of large amounts of data placing a strain on networks and data storage requirements. Communities are also unwilling to share data due to privacy policies and security concerns, thus requiring solutions for the distributed training of analytical models without any data transmission or sharing. In addition, data collected from community infrastructure often contains data biases arising from community-centric patterns that can erroneously influence a model. This would result in a loss in performance if these models are shared with other communities that do not exhibit these biases. Hence, developing an infrastructure framework that enables distributed training and sharing of models in a privacy-aware and bias agnostic manner, is a challenging but powerful solution that can significantly benefit all communities and improve their monitoring capabilities.

1.4 Driving Use Case - Drinking and Storm Water Infrastructures

Community water networks are critical infrastructure responsible for the supply, distribution, and storage of water resources. In addition to the various societal water needs, the presence of resilient water infrastructure is important to service continuity of other lifeline industries and domains such as power, sanitation, agriculture, healthcare and manufacturing, among others. As a driving use-case in this thesis, we look at two major types of water networks - drinking water distribution, and stormwater networks. Today, with widespread urban development and population growth, these water infrastructure networks have grown in scale and complexity, and in many places are aging and thus increasingly vulnerable to adverse events [8]. The global demand for water is expected to increase by $20\% - 30\%$ by 2050 [21] which will further strain our limited water resources and services. Furthermore, 60% of water across these networks is estimated to be wasted due to infrastructure failures and poor management techniques [172]. Communities are now experiencing an increase in water related outages and interruptions, and hence it is imperative to design and implement effective monitoring solutions to quickly detect adverse events and ensure the health of these different types of water infrastructure.

Drinking water distribution networks (WDNs) in communities consist of large networks of pipelines with many different components (junctions, pumps, valves, tanks, and reservoirs). The vast majority of these networks are typically underground. There are predominantly two types of adverse events that are observed in WDNs - (i) physical damage to infrastructure (e.g., pipe breaks or leaks) and (ii) water contamination events. Pipe breaks result from stress caused by factors such as corrosion, pipe displacements, extreme weather, etc, and disrupt water service to the community. Contamination events arise from the introduction of harmful contaminants like human waste, pesticides and chemicals into the water that

can then be consumed by the community resulting in public health consequences and long-lasting psychological impacts. These contaminants can enter pipelines through malicious activity or even from pipe breaks through backflows [155]. The impact of physical damages to the infrastructure and compromises to the quality of supplied water can be devastating to society and cause huge economic and public health issues such as massive flooding, outbreak of waterborne epidemics, shortages of clean drinking water, damage to property, etc [146].

Stormwater networks comprise of large and predominantly above-ground drains whose purpose is to collect water from precipitation, snow and ice melt, and surface runoff, and convey it to nearby streams, rivers, or other water bodies. With increased urbanization, there are more impervious surfaces (e.g., roads, buildings, etc.) that prevent rainfall from infiltrating into the ground. Hence, these networks are critical to prevent urban flooding and to transport runoff away from commercial and residential areas into nearby water bodies. Stormwater harvesting techniques are also important to build self-sustaining communities. However, if contaminants are present in stormwater runoff, flushing it into water bodies can destroy marine ecological systems and can also impact safe harvesting of stormwater. Garbage and human/animal waste deposited on roads, lawns, etc., can flow with rainwater into the drains, and other harmful chemical contaminants like Zinc, Copper, Lead, etc., are often illegally dumped into the storm drains by nearby industries, thereby impacting the quality of stormwater runoff [173].

Current status of water infrastructure monitoring: Today, monitoring drinking and storm water networks involves a large amount of manual effort to obtain water samples for contaminant detection (i.e., grab sampling), and deploying operators with acoustic instruments to identify pipe breaks underground. The instrumentation of IoT sensors in both types of networks is lacking, and sparse at best. WDNs have instrumented water meters primarily for billing, SCADA systems when available are deployed at pump stations above ground, and automated water quality monitoring is mainly present at storage and treatment plants. The

vast networks of pipelines or storm drains are not actively monitored, since instrumenting the entire network is prohibitively expensive to deploy/maintain, thus requiring intelligent deployment approaches. Additionally, some infrastructure locations have very intermittent network connectivity, impacting data transmission. However, the availability of affordable IoT sensors with a range of sensing modalities, coupled with powerful analytical models and compute resources presents a rich opportunity to develop effective IoT-driven monitoring solutions for water infrastructure networks.

Monitoring challenges in water infrastructure: The aforementioned challenges (Section 1.3) are critical to address for water infrastructure monitoring. These challenges are further exacerbated in water networks due to inaccessible locations (e.g., underground pipes) and need for ruggedized instrumentation. In addition, the flow of water in these networks are driven by time-varying demands (WDNs) or precipitation (stormwater), resulting in complex interdependencies which are especially pronounced in the presence of adverse events (e.g.) mixing of contaminants. Knowledge of these physical phenomena must be incorporated by monitoring frameworks to ensure effective solutions.

1.5 Thesis Contributions and Organization

This thesis aims to address some of the above challenges (Section 1.3) through cross-layer and cross-community approaches to develop a comprehensive infrastructure monitoring framework. We develop solutions that exploit the characteristics of community structure and behavior and optimize the deployment, operation, and generalization of infrastructure monitoring solutions within and across communities. This heterogeneity-aware approach limits the adverse impact of events on the community, avoids burdening resource-constrained devices and servers, adapts to changing community conditions, and supports heterogeneous monitoring components (e.g., devices, networks, analytical models). We demonstrate the ef-

fectiveness of our techniques across different infrastructure monitoring settings and testbeds.

The following is the overall organization and research contributions of this thesis:

- Chapter 2 surveys related work across the deployment, operation and generalization of infrastructure monitoring solutions.
- Chapter 3 introduces our overall approach in greater detail. We introduce a cross-layer framework for infrastructure monitoring and describe key challenges and the research problems addressed in this thesis.
- Chapter 4 describes our impact-driven approach to IoT sensor deployment for community infrastructure monitoring. Our approach leverages in-situ and mobile sensing devices, to determine deployment locations that rapidly identify and localize adverse events to minimize their impact on the community. Our key contributions include a two-phase approach to first model the vulnerability of a community based on its structure and event characteristics, and its subsequent use to determine event impact and optimal sensor placement locations. Our sensor placement approach can adaptively adjust sensing resolutions on-demand within the infrastructure, determine required sensing capabilities based on the event characteristics, and respond to varying event severities.
- Chapter 5 introduces our operational decision making framework, titled REAM, that meets the monitoring requirements of applications by adaptively selecting the optimal workflow of devices, communication networks, and analytical models based on the community infrastructure state. Our key contributions include a methodology for community stakeholders to flexibly define workflows of monitoring components, a learning-based approach to select the optimal workflow by balancing the application monitoring requirements with the resource availability on edge/cloud servers, and novel techniques to adaptively switch workflows when infrastructure conditions change.

- Chapter 6 explores the generalization of monitoring solutions from one community or location to another while maintaining performance. Our key contributions include a methodology to perform distributed training of analytical models using data from multiple communities in a privacy-aware manner, novel algorithms to ensure the generalizability of these models without being influenced by community-centric data biases, and a methodology to combine distributed training with generalization to build monitoring solutions that can be shared across communities.

- Chapter 7 concludes the thesis with the contributions made and the lessons learned, and presents future research directions to further improve community infrastructure monitoring.

Chapter 2

Related Work

In this chapter, we survey relevant work to provide an appropriate background for this thesis. We start with an overview of existing IoT system architecture designs to enable smart community infrastructure across different domains. We then discuss how IoT sensors (both in-situ and mobile) are placed or deployed for different infrastructure monitoring applications and also present a detailed overview on sensor placement in water infrastructure. Next, we explore solutions that have been proposed to improve the different operational components of an end-to-end infrastructure monitoring framework – (i.e.) managing the collection of sensor data, its transmission over different communication networks, and the subsequent analysis using analytical models to obtain actionable information. We finally review research done on distributed training of analytical models through data from multiple sources using federated learning techniques and survey prior work on handling data biases during model training to ensure that they can generalize and hence be shared across locations.

2.1 IoT Architectures for Smart Communities

IoT-enabled smart community and smart infrastructure projects have been deployed worldwide. There is an increased push by community agencies and stakeholders to use IoT technologies in applications across multiple domains. In this section, we review several of these projects across varying community-scales – ranging from personal wearable devices to large infrastructure.

At the personal level, healthcare monitoring using wearable devices has many popular applications driven by IoT systems [169, 61, 58]. People want real-time access to their health conditions like blood sugar levels, heart and pulse rates, etc, and these applications take raw sensor signals, analyze them, and provide personal health feedback. There has also been work on expanding IoT-driven architectures for healthcare to entire communities, in particular utilizing device and network usage data for monitoring the localization and spread of infection. This has become especially significant during the ongoing COVID pandemic [53, 151]. Safe Community Awareness and Alerting Network (SCALE) [14] is an affordable personal and home safety project developed at UC Irvine. Multi-sensor boxes are placed at residents' homes to provide safety-related sensing capabilities including motion, explosive gas, and personal fall detection, and applications have been developed to trigger preventive actions.

Moving up in scale, TIPPERS is a smart-building management system designed by Mehrotra et al. [97] that utilizes sensors (e.g., cameras, acoustic, RFID, etc.) in addition to wireless network traffic to support several smart-building applications such as occupancy monitoring, dynamic HVAC control, waste management, and emergency evacuations, among others. Additionally, they present a policy-driven operational framework that preserves the privacy of individuals at different layers of abstraction. Efforts have also been made at developing smart IoT-driven solutions for college campuses and small communities [36], often to ensure

the well-being and safety of residents. Tsai et al. [152] develop a monitoring framework using a collection of edge servers and smart streetlamps with devices like cameras and acoustic sensors on-board. The framework services multiple applications like illegal parking detection and pedestrian counting, and they also developed solutions for the efficient processing and storage of video streams from these streetlamps.

Initiatives to develop IoT architectures for large cities and infrastructures have gained popularity in recent years. Cenedese et al. [28] designed a real-world urban IoT system named Padova Smart City deployed in Padova, Italy. They focus on air quality and traffic applications and leverage both cloud and edge for analyzing the collected data. They also provide examples of the impact of spatio-temporal patterns on events in the community. Zhang et al. [180] developed VideoStorm, a traffic analytics framework for Seattle, USA. Here, video streams from traffic cameras across the city were collected on cloud servers, and analyzed using neural networks for applications like accident detection, license plate recognition, etc. The Community Seismic Network (CSN) [37, 74] is a participatory IoT system created by the California Institute of Technology (CalTech) to help with early alerting of earthquakes in Southern California using cheap accelerometers attached to residents' personal computers and devices. The accelerometers detect changes in acceleration and report changes to a service running on the cloud.

The above research efforts explore the potential of IoT-enabled architectures and solutions for smart communities and infrastructure. They also identify and present several challenges towards developing a comprehensive infrastructure monitoring framework (e.g., multiple diverse applications, monitoring effectiveness vs. resource-efficiency, adapting to changing conditions, etc.) that we address in this thesis.

2.2 Enabling Community Infrastructure Monitoring

A comprehensive IoT-driven infrastructure monitoring framework must be capable of supporting multiple applications with different objectives while leveraging distributed heterogeneous sensors, diverse communication networks, limited compute resources at the edge and cloud, and analytical models of varying complexity. The primary objectives of such monitoring frameworks is to provide continuous and comprehensive information about the infrastructure, to detect events in a timely manner, and to even proactively predict them. We categorize infrastructure monitoring frameworks into three phases – deployment, operation, and generalization across communities. In this section, we present an overview of existing work that has typically addressed each of these phases in isolation, hence falling short of a comprehensive end-to-end solution. We also highlight several open challenges across these phases that we address in this thesis.

2.2.1 Deployment: Sensor Placement for Data Collection

Sensor placement or deployment is fundamentally critical for any smart infrastructure monitoring solution. IoT sensors detect physical phenomena, generate formatted data, and deliver them (actively or passively) to edge and cloud servers where they are subsequently analyzed. Different community infrastructure systems present different deployment challenges (e.g., crowded smart buildings vs. underground water networks) and also require diverse monitoring applications. To support these applications, prior efforts traditionally relied on in-situ or static sensors that were installed and remained in place. However, with the rise of mobile edge devices, an increasing number of monitoring efforts are attempting to leverage the mobility advantages and develop hybrid (in-situ + mobile) sensor placement solutions. Inspired by Zhu et al. [185], Table 2.1 presents a comparison between in-situ and mobile IoT sensors for different aspects of infrastructure monitoring. We first survey efforts using these different

	In-Situ sensors	Mobile sensors
Infrastructure dependency	Leverage existing infrastructure to enable real-time monitoring	Low dependency – constant addition and removal of devices ; ability to traverse regions with low accessibility
Coverage	Typically have large range	Lower ranges ; make up with mobility
Performance	Stable, predictable, controllable since deployed in place	Dynamic with low predictability, uncertainty due to probabilistic movement
Flexibility	Low; typically cannot change functionality	High; capable for on-demand deployment
Costs	High deployment and maintenance costs but low operational cost	High operational costs (human labour to provide mobility) , often lower device costs

Table 2.1: Comparison betweeen in-situ and mobile sensors for infrastructure monitoring

types of sensors and then present an overview of sensor placement approaches that have been proposed in the literature.

In-Situ sensors

In-situ sensing is useful in community infrastructure that requires monitoring coverage of a region using a large number of devices to provide continuous monitoring of specific metrics or events. These in-situ sensors, when deployed, utilize nearby infrastructure for power supply and network access (wired or wireless). Depending on the infrastructure and the monitoring application, these devices may be ruggedized to maintain performance in harsh environmental conditions. There have been several efforts to deploy in-situ sensors for infrastructure monitoring. The Array of Things (AoT) [125] is a collaborative effort led by the University of Chicago that instruments multi-purpose environmental sensing devices on buildings, streetlamps, etc, to provide real-time, location based data about the ambient environment (air quality, noise) using a cloud platform. Farmbeats [158] is a smart agriculture project

from Microsoft Research that instruments in-situ chemical sensors (e.g., pH, moisture, etc.) and cameras on large farms to assist with precision agriculture, monitoring temperature and humidity in food storage, and monitoring animal shelters. They setup a solar powered IoT base station on the farm that uses wireless connectivity for data collection and analysis. In addition, there have also been several efforts towards instrumenting water distribution networks (WDNs) with in-situ sensors to measure network parameters like pressure, flow rate, turbidity, etc, in order to detect and localize events in a timely manner. For example, PIPENET [140] in Boston deployed a network of in-situ sensors connected wirelessly to monitor water transmission piplines by collecting hydraulic and acoustic data and WATERWISE [174] was a system deployed in Singapore to enable real-time monitoring of WDNs. WaterBox [69] was a small-scale testbed that was developed to test system control algorithms in a fail-safe environment.

Mobile sensors

The use of mobile sensing devices for monitoring has gained significant popularity and are being used by infrastructure monitoring applications across several domains. The flexibility (on-demand deployment) as well as the ability to reach otherwise inaccessible locations present significant advantages over in-situ sensors. Ambicity [168, 167] is a crowdsensing project from Inria which uses the microphones on participants' smartphones to perform noise and air pollution sensing while they move around the city of Paris. Based on the collected data, the associated cloud service creates real-time noise and air pollution maps. Mosaic [41] is a mobile sensing project from Zhejiang University that uses sensors mounted on city buses to create city-scale fine-grained maps for PM 2.5 (particulate matter, an air pollution indicator). Rahman et al. [120] developed BreathEasy, that used multimodal sensors embedded in consumer mobile devices for non-invasive, low-effort respiratory assessment. Drones are also being used as mobile sensing devices for applications such as wildfire

detection, traffic monitoring, and smart agriculture [9, 71, 158]. These efforts showcase the advantages of using mobile sensors to collect data across vast areas with low costs. In water networks, to tackle the issue of inaccessible locations and underground pipelines, the use of mobile sensors has been gaining popularity in water infrastructure as well. Systems like SmartBall [48] and PipeProbe [79] drop mobile sensors into the network which traverse and monitor pipes by moving with the water flow. These mobile sensors flow along with the water through the pipes and have acoustic and chemical sensors on board to detect events during traversal. They typically are deployed and collected through water network infrastructures like hydrants or manhole covers, and data is accessed post-collection and transmitted using wireless networks above-ground.

Sensor placement approaches

While utilizing in-situ and mobile IoT sensors can help with real-time monitoring and event detection, community agencies and stakeholders are often limited by budget and operational costs, and hence require strategies for intelligent placement of a limited number of sensors to ensure effective infrastructure monitoring. Most approaches for sensor placement assume a typical setting, where the objective is to maximize the coverage and event detection likelihood given a budget. Such approaches [40, 62] assume that every sensor has a sensing range around it, and an effective deployment uses the least number of sensors (or till the budget is exhausted) required for total coverage of the infrastructure while minimizing the overlap of sensing ranges. Other efforts to combine deployment of in-situ and mobile sensors optimize placement using Lyapunov functions [186], Probabilistic Graphical Models (PGM) [95], and other structural-quality heuristics like inter-node distance, network connectivity, etc., as summarized in [178].

Sensor placement approaches for water networks typically convert the infrastructure into a graph, where nodes represent junctions and links represent piplines. Typical heuristics

aim to optimize factors such as event detection time, likelihood of detection, etc. These include graph theoretic approaches [78, 70] that utilize shortest path and set cover variants as well as optimization methods using mixed integer programming [19, 18] that typically work for small networks. Challenges such as the Battle of Water Sensor Networks (BWSN) [107] have resulted in more efficient approximation methods for placement [76] that scale to larger networks. Typical leak detection techniques use network coverage as the prime objective to design placement heuristics. Deterministic methods like Branch-and-Bound can guarantee optimal solutions [132] for limited scale. Techniques utilizing genetic algorithms can scale [115], but have long runtimes and are hard to tune, and several approximate solution alternatives have been proposed [117, 113]. Existing work on combining mobile and in-situ sensor deployments in water networks typically assume the prior placement of static infastructure - either static sensors [112, 106, 122] that cover a predetermined portion of the network, or sink nodes/beacons [143, 43] in the junctions that communicate with the mobile sensors, with the objective of determining the number of mobile sensors needed and their release locations to cover the network. However, we argue that it is essential to consider the deployment of both types of sensors simultaneously since the placement of one type directly affects the performance of the other. Also, these prior approaches assume that all events are uniform and do not distinguish between them. We address these open deployment challenges in this thesis.

2.2.2 Operation: Decision Making for Infrastructure Monitoring

The next steps in infrastructure monitoring, once sensors have been deployed, is to transmit their data streams over different networks (wired, wireless) to edge and cloud servers where they are passed as input to analytical models (Figure 1.4). Hence, an infrastructure monitoring framework operates pipelines of sensors, network links, and analytical models that together deliver actionable information. Each component of this pipeline can consist

of multiple options (e.g., using a camera vs. acoustic sensor, or Bluetooth vs. WiFi) and hence, each monitoring application could be serviced by multiple possible pipelines. Each pipeline of sensors, networks, and analytical models provide a certain benefit to the application (e.g., accuracy) while incurring resource costs like power consumption. In addition, the benefit provided by a pipeline also depends on community conditions. For instance, the performance of a camera could be impacted by rain obscuring its lens. Hence, a good monitoring framework needs an adaptive decision-making solution for these operational choices, in order to achieve high monitoring performance while judiciously using limited resources to support multiple applications. In this section, we present an overview of existing work, where research has typically focused on optimizing different parts of this monitoring pipeline.

URMILA, developed by Shekhar et al. [134] is a middleware solution to manage resources across the cloud, fog and edge to ensure that SLO violations are minimized for latency-sensitive IoT applications, particularly those that are utilized in mobile environments. They propose approaches to predict network latency and energy consumption of applications, and select the most suitable server to execute each application. Alhassoun et al. [5] propose SAFER, an energy-aware perpetual home IoT system where battery-operated and wall-powered IoT devices co-execute to ensure the safety of occupants. They use a semantic approach that extracts activities-of-daily-living (ADL) from device data to drive energy-optimized sensor activations. Vaisenberg et al. [156] leveraged Partially Observable Markov Decision Processes (POMDP) to control surveillance cameras to record events in resource-constrained smart spaces. They used PTZ cameras in a smart building which had resource constraints, and proposed an approach using POMDP that could predict future states in which the events are likely to occur, based on partially-observed past states, thereby allowing them to proactively activate and deactivate the cameras to maintain good performance while conserving resources. Nesa et al. [103] analyze network topology in terms of relative distances and link qualities between sensors, as well as the remaining battery life of the sensors and develop a sensor ranking algorithm. Based on this ranking, a subset of sensors are activated

to conserve overall energy consumption. A similar objective using network topology was achieved by Du et al. [44] for water distribution networks (WDNs), wherein they used a dynamic programming approach instead.

DeepDecision [121] is a framework by Ran et al. for mobile video analytics, that leverages deep learning models on both mobile edge devices and cloud servers to make offloading decisions. They perform extensive measurements to understand the tradeoffs between video quality, network conditions, battery consumption, processing delay, and model accuracy, and use this to choose where, and which deep learning model to run under variable network conditions. Another approach that also uses resource profiling for video analytics is VideoStorm by Zhang et al. [180] who profile the tradeoff between video quality vs. resource consumption, and consider the unique requirements of different applications in terms of latency and quality to take decisions on the different analytics to run on the same video stream. Benson et al. propose FireDeX [15], a middleware to manage prioritized delivery of critical data from IoT sensors. Their approach accurately estimates end-to-end performance metrics (e.g. delays, success rates) across different network links and selects the best path to transmit important data. This leverages their earlier work [17] that presented an approach to gather network-awareness via a resource-aware adaptive probing mechanism and dynamically redirecting IoT data flows. They leverage these network-cognizant decision making approaches for emergency response applications.

Decision making approaches with Markov Decision Processes (MDPs) and Reinforcement Learning (RL) have also been proposed for different smart community applications. In these situations, an agent is trained to learn an optimal policy, which maps community states to optimal actions that must be taken. Pettet et al. [116] use MDPs for dynamic resource allocation in emergency response systems, where the goal is to optimize ambulance locations to minimize response times to emergency calls while considering constraints on the number of available ambulances and the locations where they can be stationed (i.e., variant of the

sensor placement problem). Han et al. [56] used RL to develop a middleware for event identification in community water networks. The agent's goal is to select optimal locations for grab-sampling given existing observations to balance the tradeoff between human effort (large number of locations) vs. missing events (low number of locations). Mao et al. [93] used RL to develop a task packing framework that handles resource demands from multiple applications on cloud servers. The agent learns to optimize objectives like application latency using cluster assignments by looking at past historical demands. Similarly, Gai et al. [49] leveraged RL to allocate analytics workload among edge servers, where the agent considers both energy costs and analytics response time as metrics to optimize for. Chen et al. [32] adopted RL algorithms to migrate edge analytics between servers for better energy efficiency or service quality. Their Q-learning based algorithm was evaluated on a testbed with two edge servers and four mobile devices.

2.2.3 Generalization: Sharing Solutions Across Communities

The ability to share data and models across communities can provide significant benefits. Data sharing enables the training of more robust models that can detect a wide and diverse set of events, while model sharing can enable communities to reuse solutions instead of needing significant monetary and time investment to develop their own. Sharing can also help bridge the gap in infrastructure monitoring quality between resource-poor and resource-rich communities. However, model sharing can be limited by data biases, which can be present in a community's data due to local patterns like the weather that do not have a causal relationship with the observed events. Reliance on these biases can negatively impact the performance of models when deployed in new communities where they are not present, and hence solutions to train generalizable models that ignore biases is powerful. Data sharing to train robust models, on the other hand, is also challenging due to increasingly stringent privacy policies and security concerns, wherein communities and users are unwilling

to share sensitive data. To address this, distributed model training approaches like federated learning [96] are gaining importance to leverage the benefits of diverse data collections from distributed sources. In this section, we present an overview of prior research work on training generalizable models as well as federated learning approaches.

Generalization of machine learning models

Prior work on machine learning generalization have predominantly looked at problems in computer vision. There have not been any significant efforts made to train generalizable models for smart community applications where data biases and spurious correlations are often present. Some initial work has been proposed in the space of domain adaptation for smart community applications [176, 111, 183, 89] where models trained on one data distribution are quickly retrained for new distributions. However, they assume that the new test data distribution is already known, unlike the harder problem of generalization where the model is expected to identify the causal features and perform well on any unseen test distribution. Hence, training generalizable models for community infrastructure monitoring is an open challenge that we have addressed in this thesis.

There are various approaches that have been proposed to improve the generalization of machine learning models for computer vision applications. Causal model discovery [109, 59] aims to find an underlying causal graph to obtain an invariant feature set that is a causal predictor of the target. Arjovsky et al. [7] propose Invariant Risk Minimization that estimates an invariant model optimizer across different distributions. Data augmentation techniques are also popular and aim to make the model more robust by training using instances obtained from neighbouring domains hallucinated from the training domains, and thus make the network ready for these neighbouring domains. Shankar et al. [133] augment data using instances perturbed along directions of domain change and use a second classifier to capture this. Volpi et al. [170] apply this to single domain data, while Carlucci et al.

[27] apply augmentation to images during training by simultaneously solving an auxiliary unsupervised jigsaw puzzle alongside.

Decomposition based approaches represent the parameters of the network as the sum of a common parameter and domain-specific parameters during training [39]. Khosla et al. [72] applied decomposition to domain generalization by retaining only the common parameter for inference. Li et al. [83] extended this work to Convolutional Neural Networks (CNNs) where each layer of the network was decomposed into common and specific low-rank components. Piratla et al. [118] recently proposed a more efficient approach that decomposes only the last layer, imposes loss on both the common and domain-specific parameters, and constrains the two parts to be orthogonal. Another approach is to pose the generalization problem as a meta-learning task, whereby we update parameters using meta-train loss but simultaneously minimizing meta-test loss [82]. Prior work on meta-learning has been studied either in the context of few-shot supervised learning methods which adapt using small amounts of labeled data from the new domain [131, 123, 47], distribution shifts in only test domains [42, 181], or only considering label shifts [90, 141].

Other approaches include adversarially learning representations that are invariant with respect to domain-specific features using perturbations [4, 153] as well as domain erasure methods which estimate features that have the same distribution across different domains using techniques like data-reconstruction, projection, MMD, etc [50, 52, 84]. There have also been some prior work for other applications like visual question answering [145], business process predictions [164] and medical diagnosis using human annotated spurious features [139].

Federated learning

As described in Section 1.1, IoT data generation for different community applications has exploded in recent times. Training analytical models in a centralized manner by collecting and transmitting data from different sources to a single server is challenging due to the strain on network bandwidth and high storage requirements. Additionally, there has also been a rise in privacy and security concerns such as the leakage of confidential personal data [184] and ransomware attacks on community infrastructure like water networks and the smart grid [177] resulting in agencies and stakeholders being unwilling to share their data. Federated learning (FL) has emerged as a paradigm to address these concerns, where distributed devices or edge servers can collaboratively train a shared global model without the need for data transmission or sharing.

There have been several efforts to utilize federated learning for smart community applications. Liu et al. [91] use FL to predict traffic flows by training models directly at the edge (i.e.) on-board vehicles, using attributes such as road geometry and the weather. They show that distributed training by leveraging data from many vehicles helps provide better traffic prediction outcomes. Similarly, Samarakoon et al. [129] propose an FL-based approach to achieve ultra-reliable low latency communications in vehicles. Lyapunov optimization is used to calculate the joint power and resource allocations to enable low latency communication for vehicular users. Federated learning has also been leveraged by several efforts to train accurate models for medical diagnosis using sensor data, while preserving patient privacy [126, 24, 135]. Nishio et al. [104] and Xu et al. [175] propose approaches to use FL on resource-constrained edge devices, to improve efficiency by considering the varying levels of computation capabilities in different IoT devices.

The majority of existing work on using federated learning for smart community applications leverage the FedAvg algorithm proposed by McMahan et al. [96]. It trains a global central-

ized model by periodically averaging the weights of local models. Improvements on FedAvg have looked at reducing the communication costs such as FedMA [171], CMFL [92], and ASO-Fed [33]. However, FedAvg has been shown to perform poorly, or in some cases even diverge empirically in settings where the data is non-identically distributed (i.e., non-iid) across devices. However, the presence of non-iid data is common in smart community applications, since sensors and devices can (i) have differing numbers of data samples due to sampling rate, device runtime, etc. and (ii) have differing data distributions due to differences in geographical issues, weather patterns, etc. To address this issue, several approaches have been proposed including FedProx [83] by Li et al. where a proximal term is added to the client loss functions, thereby limiting the impact of local updates by keeping them close to the global model. Agnostic Federated Learning (AFL) [100] proposed by Mohri et al. is another improvement which optimizes the central model for any new distribution that is a mixture of the local client distributions. Karimireddy et al. propose SCAFFOLD [68] which uses control variates (variance reduction) to correct for any distribution drift or changes in the local clients. However, none of these approaches consider the presence of data biases or spurious correlations in the training data on local devices. Hence, developing a generalizable federated training approach for effective community infrastructure monitoring is critical, and is an important contribution of this thesis.

Chapter 3

Approach Overview

In this chapter, we present our overall approach to understand and address challenges in developing IoT-driven infrastructure monitoring solutions for communities. We detail a cross-layer approach to infrastructure monitoring and discuss our past projects like SCALE and AquaSCALE, that were developed for enabling and improving the resilience of different monitoring layers during adverse events in the community. We then identify major research challenges to move forward towards developing comprehensive infrastructure monitoring frameworks that leverage community structure and behavior and which can be deployed across communities. We finally present an overview of our solution approaches to address these problems and highlight our cross-layer and cross-community contributions in this thesis.

3.1 Infrastructure Monitoring

There have been numerous initiatives like the SmartAmerica Challenge [30], to encourage industry, academia, and government agencies to develop solutions that leverage IoT and related technologies for infrastructure monitoring. Their vision for a comprehensive infrastructure monitoring framework involves effective monitoring, while being able to handle multiple infrastructures and monitoring applications, across different communities. The fundamental flow of data and information in such a framework is depicted in Figure 3.1, where data is generated by sensors at the lowest layer and is transmitted through communication networks and analyzed by models to finally service application objectives. Following these initiatives, our group at the University of California, Irvine (UCI) has collaborated with several industry and government partners to develop solutions to optimize different components of this monitoring workflow. A vast majority of our work thus far has focused on developing solutions to improve the resilience of infrastructure monitoring when different components of the monitoring workflow fail due to adverse events.

The SCALE project [14] was created to address challenges related to improving the resilience of monitoring frameworks. Reliable delivery of data from devices to edge and cloud servers is critical for infrastructure monitoring, to ensure that no events are missed. However, network outages can happen due to various reasons including large scale events like earthquakes. To address this, we developed several approaches to enable reliable data delivery and exchange under various network failures. We proposed GeoCRON [16] as a solution to achieve reliable data delivery during geo-correlated infrastructure failures, by exploiting geographically redundant network routes to avoid failures. It uses information of the underlying routing infrastructure to establish multiple geo-diverse routes in the network overlay and sends multiple copies of the data along these routes to improve the chances of successful data delivery during network failures. We further developed a middleware for resilient IoT data exchange (RIDE) [17] that used Software-Defined Networking (SDN) to redirect data flows between

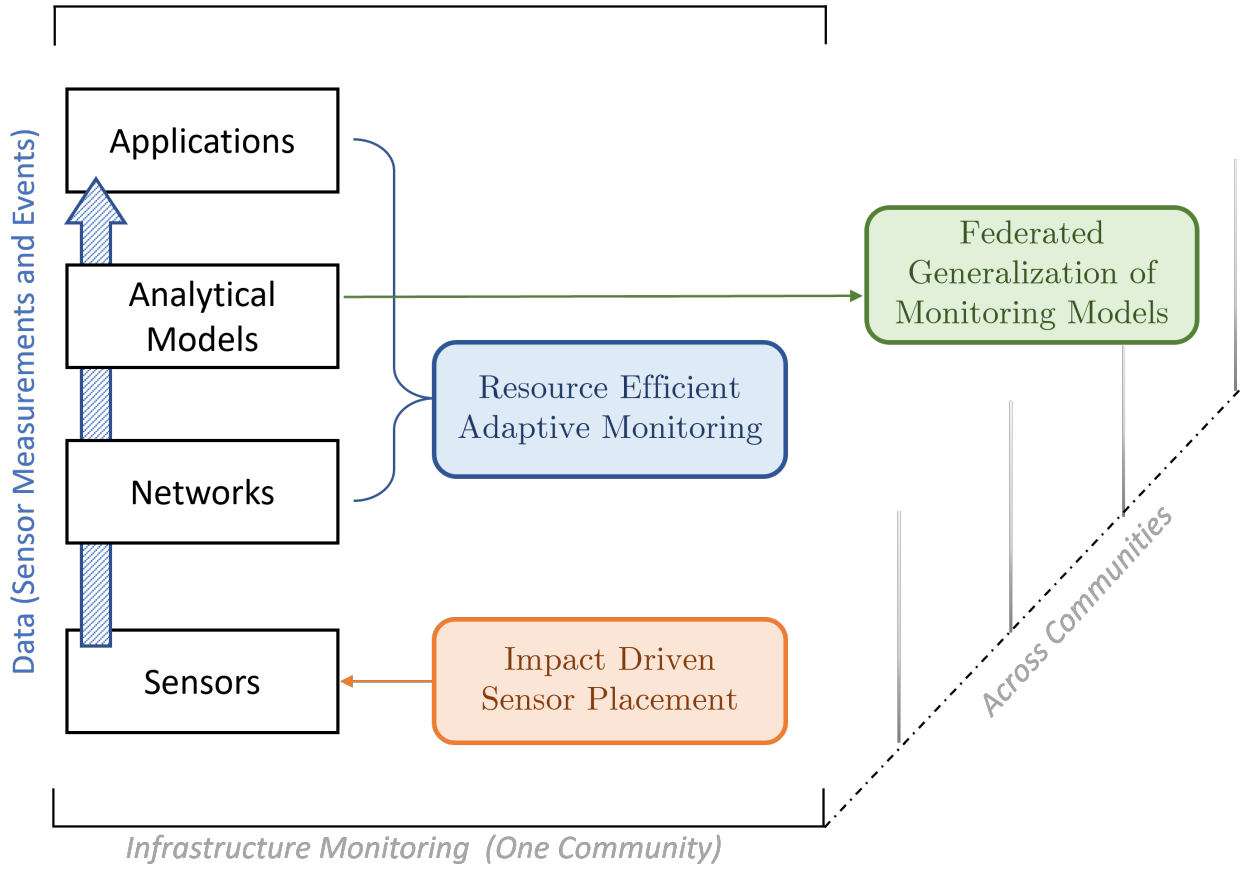


Figure 3.1: Flow of information in infrastructure monitoring and our cross-layer and cross-community contributions in this thesis

edge and cloud servers during network failures or congestion.

Following the success of SCALE, two projects – AquaSCALE [56, 57] and SWADE [166, 159], were proposed to develop infrastructure monitoring solutions specifically for water networks. AquaSCALE focused on drinking water distribution networks (WDNs) and proposed solutions to leverage IoT-driven infrastructure measurements along with human inputs (e.g.) social media, to explore three key resilience problems – (i) identification and isolation of concurrent pipe failures, (ii) state estimation of WDNs under extreme events like earthquakes, and (iii) contaminant source identification using human-in-the-loop based sensing. SWADE is an ongoing project that goes beyond drinking water networks and also looks stormwater and wastewater infrastructures. Currently, each of these water infrastructure systems

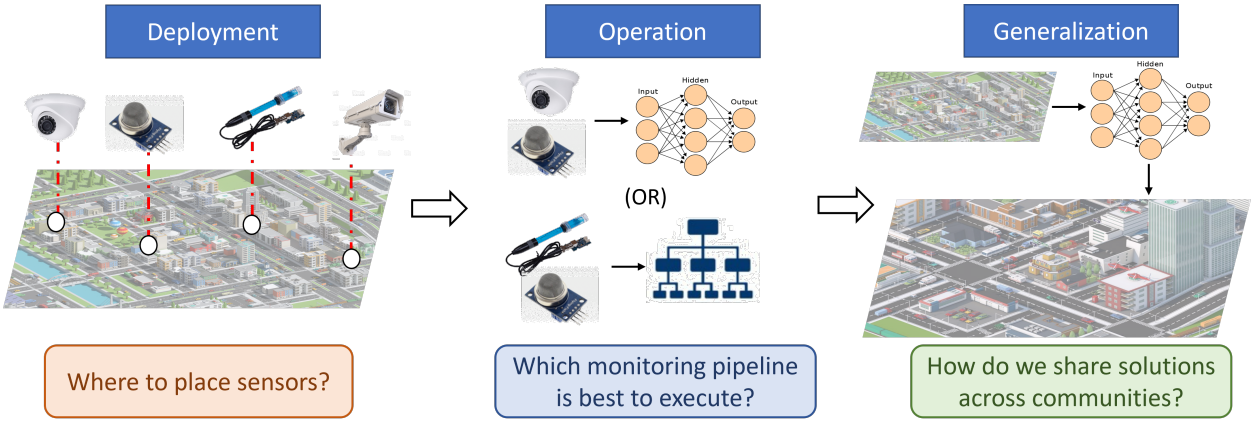


Figure 3.2: Key questions addressed across deployment, operation, and generalization of infrastructure monitoring frameworks

operates independently despite common objectives and overlapping information. Hence, developing and sharing monitoring solutions across these different types of infrastructure that can aggregate data and structural information to leverage commonalities can greatly improve planning and operational procedures.

The work in this thesis augments and builds upon these earlier efforts and Figure 3.1 depicts our cross-layer and cross-community contributions. We address challenges not only across all the layers of an infrastructure monitoring framework but also challenges arising from the need to extend and share single-community solutions across communities. The approaches developed in this thesis present a significant step forward towards comprehensive infrastructure monitoring across communities. Specifically, we look to answer three key questions across the deployment, operation, and generalization of infrastructure monitoring frameworks as depicted in Figure 3.2. We address community infrastructure monitoring challenges pertaining to effective sensor deployment, efficient monitoring operation, and the generalization of monitoring solutions across communities. In the rest of this chapter, we discuss challenges that need to be addressed to answer these questions and present an overview of our solution approaches.

Figure 3.3: Diversity in structure and behavior across three communities

3.2 Research Challenges

3.2.1 Diversity in Community Structure and Behavior

A major challenge towards developing a comprehensive infrastructure monitoring framework is to handle the inherent diversity present among communities. We classify this diversity into two aspects - structural and behavioral, and we use the example of three different communities (A, B, C) in Figure 3.3 to illustrate some of this diversity. Each row in the figure corresponds to four images taken at different times of the year at the same location in the corresponding community.

Structural differences between community infrastructure can manifest in terms of the underlying physical topography, the layout and locations of infrastructure, population demographics impacted by events, scale of operation as well as available resources (capital, instrumentation, data), among others. To illustrate this, consider an example use-case application of traffic monitoring (pedestrian and vehicular) across the three communities in Figure 3.3. We observe that A represents a busy intersection, while B depicts a small road, and C shows a

pedestrian sidewalk. These three different community settings present significantly different flows of pedestrians and vehicles, where we can expect the largest number of vehicles in A, followed by B, and no vehicles in C, while the number of pedestrians would typically be largest in C. Due to these differences, the sensor instrumentation for monitoring would also differ across these communities in terms of the number of cameras required, their degree of coverage, and the extent of monitoring needed. This example shows how structural differences across communities can impact monitoring requirements even if the application or type of infrastructure is the same.

Behavioral differences can occur due to differences in weather phenomena, temporal patterns, correlated occurrences of events, unique community events like festivals, etc. From Figure 3.3, we see that weather events like snow and rain, can impact the behavior of traffic and pedestrians. For instance, the number of pedestrians could reduce, and the speed of traffic movement could also change. The weather could also impact the quality of images that are obtained from the camera, thereby affecting the performance of analytical models relying on these images. This degradation in image clarity can also happen due to day/night time differences (like in community C), where the temporal changes also cause a difference in the number of pedestrians observed in the images, showing that these behavioral differences not only impact infrastructure monitoring across communities, but also within a community.

Existing infrastructure monitoring efforts focus on individual applications in specific communities. If we are to develop a more comprehensive monitoring framework that can be deployed across communities for different applications, we must address and leverage this diversity in community structure and behavior.

3.2.2 Operational Heterogeneity

The presence of heterogeneity across the different layers of a monitoring workflow (Figure 3.1) can create challenges during operation or at run-time. Effective workflows depend on the kinds of applications, analytical models, network links, devices and data types that are being supported, and each of these components often exhibit heterogeneity that can impact the operation of infrastructure monitoring.

The heterogeneity in the types of IoT sensors or devices is well documented, ranging from analog sensors like acoustic, temperature, air quality, etc, to more complex devices like cameras and mobile phones. Each of these devices provides specific measurements or data features (e.g., cameras provide images), thereby limiting their use to specific monitoring applications. Even within specific types of devices, like cameras, we can observe heterogeneity in terms of the sensing range, effectiveness, monetary costs, as well as resource consumption between different devices of the same type. This subsequently results in heterogeneity in the data being collected, not just in terms of the type of data (e.g., images vs. time-series), but also in terms of data generation (e.g., bursting, periodic, streaming), data quantity and quality which can all affect the effectiveness of a monitoring workflow.

The network links on which data from sensors are transmitted can also present heterogeneous characteristics. While wireless networks are extremely popular for smart community applications, there are numerous kinds, including WiFi, Bluetooth, Cellular, ZigBee, LoRa, etc. Each of these networks provide different benefits and shortcomings for criteria like bandwidth, data rates, power consumption, and network availability, which must all be considered in conjunction with the monitoring application objectives, when determining the network links to utilize. Similarly, monitoring models that are trained to analyze the data from the sensors for different applications also exhibit heterogeneity. Machine learning has emerged as a powerful tool to leverage the vast quantities of available data to identify patterns within

infrastructure systems that could be used to identify or predict events. But there are numerous types of models that can be trained which vary in their architecture, objective (e.g., classification vs. regression), performance accuracy, and resource consumption (e.g., power, memory). The optimal model choice is dependent on individual application objectives as well as the resource availability on the servers where the model will be deployed. Finally, this heterogeneity builds up to the types of monitoring applications that infrastructures require. With the increased awareness of the importance of infrastructure monitoring, numerous applications like pollution monitoring, traffic surveillance, intruder detection, failure detection, water quality monitoring, etc. are required for different types of infrastructure. Each of these differ in their objectives, data requirements, sensitivity to network or device failures, among others.

We believe that the design of a comprehensive infrastructure monitoring framework must appropriately address the different kinds of heterogeneity described above in a real-time adaptive manner.

3.2.3 Distributed Training and Generalization

The diversity in community structure and behavior can result in the occurrence of events of different types across communities. Training analytical models on data shared by different communities can improve their robustness and capability to detect a wider range of diverse events (e.g., contamination, accidents, intruders). However, centralized training requires significant costs associated with data transmission and storage, and can place a strain on the communication networks that need to service other applications as well, thus necessitating distributed training approaches. Moreover, communities are often unwilling to share sensitive data due to privacy and security concerns, which can be addressed with distributed training. Additionally, communities often have access to differing levels of resources (e.g., monetary

capital, compute, sensing), that often result in resource-rich communities being able to train more effective monitoring models for their infrastructure as compared to resource-poor communities, where this difference could be offset by training models that can generalize or be shared between communities. However, data biases often exist in local community data which can influence the analytical models, and result in poor performance if shared with a new community where these biases are not present. Hence, distributed training and improving model generalization by handling data biases are important challenges that must be addressed to enable cross-community sharing.

Traditional approaches to training analytical models require transmission of vast quantities of data to a central server where the models are trained. Constant transmission can take up most of the network bandwidth, reducing the availability for other applications and can result in performance issues due to dropped packets. Centralized approaches also need significant storage resources to hold the large amounts of sensor data that are generated from different infrastructure, and can quickly run out of space. This also results in all the data being stored at a single location, which becomes a critical point of failure or malicious activity since monitoring community infrastructure often requires the collection of personal and confidential data. For instance, smart building monitoring frameworks like TIPPERS [97], collect WiFi usage data of occupants, while larger agencies like Orange County Public Works (OCPW) have vast amounts of data and sensitive information about water distribution networks servicing millions of people. If people's personal data or confidential and critical infrastructure information were to fall in the wrong hands, it could cause significant adverse impact to individuals and the community at large. Additionally, the number of cyber-attacks on community infrastructure have increased in recent years. A recent attack targeted the Oldsmar Water Treatment System in Florida [150], where the hacker accessed their internal systems to contaminate the water treatment by increasing the levels of sodium hydroxide from 100 to $11,000$ ppm which could have caused severe health issues if consumed. There have also been attacks to sabotage power grids, that could have resulted in widespread

power outages [149].

The goal of every monitoring model is to capture the underlying causal relationship between different features or data variables and the target event. For example, a model for detecting air pollution needs to learn the causal relationship between PM2.5 measurements (particulate matter) and whether or not air pollution exists. However, the inherent characteristics of a specific location or community can often inject spurious biases or correlations between features and the event. These spurious correlations can occur due to sensor locations, sampling issues, and other community-centric characteristics like weather patterns and so on. These correlations have no causal relationship with the event outside of the collected data (i.e.) correlation not implying causation, but because the model relies on all the correlations present in the data during training, it learns and relies on these spurious relationships. This results in the model performing poorly when given any new data, from either the same community or when deployed in another community, that does not contain these data biases.

Addressing these challenges requires solutions that can train generalizable models in a distributed manner such that they can also be deployed or shared across communities without any significant loss in performance.

3.3 Solution Approaches

In order to address these different challenges towards achieving a comprehensive infrastructure monitoring framework, we develop cross-layer and cross-community solutions as shown in Figure 3.1. This section provides a brief overview of our approaches to address three challenges – effective sensor deployment, efficient monitoring operation, and distributed model generalization. Further details of our approaches are provided in subsequent chapters.

3.3.1 Sensor Deployments to Minimize Community Impact

The goal of any distributed deployment of sensors is to provide sufficient information about the infrastructure so as to quickly capture events, particularly adverse ones, thereby mitigating their potential impact on the community. Traditionally, sensor placement approaches have focused on heuristics like coverage, and distance between sensors. However, given the diversity in community structure and behavior that we discussed in the previous section, such approaches may result in deployments that are too slow to detect high impact or harmful events, since they assume that (i) all locations in the infrastructure have the same importance, and (ii) all events are equal, both of which are often untrue.

Defining the impact of any event in community infrastructure requires several pieces of information that we look at with an example of water pipe failures: (a) infrastructure layout and properties – pipe failures in a central junction servicing large areas is more impactful than one at the edge of the network, (b) event characteristics – several large pipe bursts are more impactful than a small leak, and (c) community structure – a pipe burst in an urban centre is more impactful on the community than one in the middle of a wasteland.

Hence, determining the optimal locations to place a limited number of sensors should utilize knowledge of the community, infrastructure, and the events that need to be captured. We propose an impact-driven approach for sensor placement, where we first quantify the notion of event impact on a given community by looking at the above three aspects, and leverage this knowledge to determine optimal sensor locations that will ensure the quick identification of any high impact events so as to minimize their impact on the community. Details of our approach will be presented in Chapter 4.

3.3.2 Resource Efficient Adaptive Monitoring

In this effort, we address the operational challenges of infrastructure monitoring frameworks that arise due to the various forms of heterogeneity across the different layers that was described in the previous section. We observed that heterogeneity in monitoring can arise from devices, data, communication networks, and analytical models which all have to work together in harmony to provide effective monitoring solutions. Our goal is to develop an operational framework that can handle this heterogeneity and provide a way for users and stakeholders to deploy multiple monitoring applications with a variety of sensing, analytics and communication options, such that our framework continuously selects the optimal monitoring pipeline for each application in real-time to achieve the application objectives.

The optimal choice of sensors, analytical models and network links can constantly vary depending the community conditions (e.g., day vs night, raining vs. sunny) and hence a one-size-fits-all approach to determining monitoring pipelines will prove ineffective. Moreover, with the need to service multiple applications, it is critical to develop solutions that judiciously use the limited available resources like power, computation, storage, etc. We leverage reinforcement learning to train agents to learn different patterns in a community to then identify the best possible monitoring pipeline or workflow to use at any given time for the current conditions.

Our key idea is to balance the monitoring effectiveness of a pipeline and its associated costs while meeting both the application requirements and the resource availability. We do this by first enabling community stakeholders to define multiple monitoring pipelines consisting of sensors, network links, and analytical models for each application, and then collecting information about the community states that can influence an application like the weather. This is used to train the reinforcement learning agents to adaptively identify (in near real-time) the best operational decisions to take given the current community state information,

available resources, and the effectiveness and costs of the monitoring pipelines. We present further details of this resource efficient adaptive monitoring framework in Chapter 5.

3.3.3 Distributed Training of Generalizable Monitoring Models

In this effort, we aim to achieve our vision of training generalizable monitoring models that are robust and which can be deployed or shared across communities without any loss in performance, while addressing the associated challenges of (i) distributed training, and (ii) data biases or spurious correlations.

We achieve distributed training and also overcome the privacy concerns associated with sharing sensitive infrastructure data to train models by leveraging the federated learning paradigm. Federated learning, in contrast to centralized training where raw data is shared to a central server to train a model, trains models in a distributed manner. Each community or location instantiates its own local model that trains securely on local data, and periodically all the local models from the different communities share their model parameters (and not any data) with a central model that aggregates these parameters and transmits back the new aggregated model. Over time, this approach results in all the local models converging, essentially reflecting them being trained on all the local datasets across the different communities, but without any sharing of local data, thereby improving their robustness while also preserving data privacy.

However, while federated learning addresses the need for distributed training, this approach does not handle biases or spurious correlations in the data. We present an approach that improves upon the standard federated learning paradigm by training models to identify and ignore biases in the data in a distributed manner. We develop a set of masks, one for each input data feature, that leverages feature stability to identify biased and causal features. While training each local model, we update its local masks based on this stability

to emphasize causal features and suppress biases. During the aggregation of local models, we also aggregate these masks, and over time this allows the global model to identify and ignore biases across all the local datasets and therefore results in models that can generalize. We present further details of our approach in Chapter 6.

Chapter 4

Impact Driven Placement for Adaptive Monitoring

We begin our efforts towards developing a comprehensive infrastructure monitoring framework from the first layer – the deployment or placement of sensors. Sensor deployments are typically distributed over the community infrastructure and can consist of heterogeneous devices that can be in-situ (static) or mobile. The effectiveness of a distributed placement of sensors lies in its ability to quickly detect a wide range of events in the infrastructure. However, as we described previously, there is significant heterogeneity in the structure of communities (e.g., scale, topography, location of critical infrastructure) as well as event characteristics (e.g., event type, locations, intensity), which present additional challenges that need to be addressed.

In this chapter, we combine the benefits of in-situ and mobile sensing with various geosocial factors to develop a cost-effective hybrid sensor placement approach that minimizes the impact of adverse events on the community infrastructure. Our focus in this chapter is on water distribution networks (WDNs), but our approach can be easily extended to other

types of infrastructure as well. Our sensor placement approach can adaptively adjust sensing resolutions on-demand within the infrastructure, determine required sensing capabilities based on the event characteristics, and respond to varying event severities. We propose a two-phase planning and deployment approach that first integrates network structure, event, and community information with simulation based analytics to determine locations to install in-situ sensors and mobile sensor insertion infrastructure. We then incorporate network flow information to determine mobile sensor deployment locations and volume to quickly localize detected events to minimize their impact. Our results indicate that the proposed approach results in a placement of sensors that can quickly detect high priority events thereby minimizing any adverse impact on the community.

4.1 Chapter Overview

Water distribution networks (WDNs) constitute one of the most critical urban infrastructures and are an important community lifeline. The monitoring of water networks is essential to ensure the availability of sufficient quantity and quality of water. Today's water networks are often decades old, and their growing scale and complexity make them increasingly vulnerable to adverse events [8]. Large pipe failures or leaks and the introduction of contaminants are the most common events affecting the quantity and quality of water in WDNs. Like many other community infrastructures, water distribution networks consist of multiple interconnected components, where such adverse events can cause significant disruption to water services. Pipe leaks or failures can result from stress caused by factors such as corrosion, pipe displacements, extreme weather, disaster events, etc. Approximately $14\% - -18\%$ of treated drinking water in the U.S. is lost because of leaks or breaks in faulty pipelines [101, 88]. The quality of water in these networks can also be compromised via contaminant introduction and propagation (e.g., nitrates, metals, pesticides) through the pipes. Oftentimes, pipe breaks result in contaminants entering the pipelines through backflows [155]. The impact of physical damages to the infrastructure and compromises to the quality of supplied water can be devastating to society and cause huge economic and public health issues such as massive flooding, outbreak of waterborne epidemics, shortages of clean drinking water, damage to property, etc. Having an effective deployment of sensors in place to monitor water networks is thus essential to localize and resolve these adverse events, in particular, those that disrupt and impact the community at large.

There have been several efforts towards instrumenting water distribution networks with sensors to detect and localize events in a timely manner. Some systems like PIPENET [140], WaterWise [174], WaterBox [69], and AquaSCALE [57], install in-situ or static sensors to measure network parameters like pressure, flow rate, turbidity, etc to detect the occurrence of events. Figure 4.1(a) shows the WaterWise multi-sensor probe that holds several commercial-

off-the-shelf sensors for hydraulics and is inserted into the flow on pressured pipes. Static sensors provide continuous monitoring with one time installation and continuous communication costs. High-end static sensors provide deterministic performance, larger sensing ranges, and good accuracy. However, they are expensive and the instrumentation of civic water infrastructures at large would require significant investments (millions of dollars).

On the other hand, systems like SmartBall [48] and PipeProbe [79] drop mobile sensors into the network which traverse and monitor pipes by moving with the water flow. Figure 4.1(b) shows the deployment of a SmartBall into a water pipe. Mobile sensors detect events by traversing near them while flowing through the pipes and incur operational costs during their deployment. Operating mobile sensors typically incurs lower cost than trenching and installing static sensors to retrofit existing pipe networks. Mobile sensors also allow for adaptive sensing on-demand as sensors can be deployed at different locations, at different times, and with different sensing capabilities based on the need. However, they do not provide continuous monitoring and have low sensing ranges. They also require a larger number of sensors since they have probabilistic movement through network junctions, and also need infrastructure support for their insertion into the network.

We argue that an intelligent sensor placement methodology is important for rapid and cost effective identification of events throughout infrastructure networks. In contrast to current placement approaches that are coverage based and treat all events uniformly, our approach is based on identifying the needs of the community and the impact of every event. There are three key observations that drive our impact-driven approach to instrumenting water networks:

1. First, while detecting all events is important, **not all events are equally impactful**. For instance, a pipe break affecting a hospital is more critical to identify rapidly as compared to one that is affecting a household.

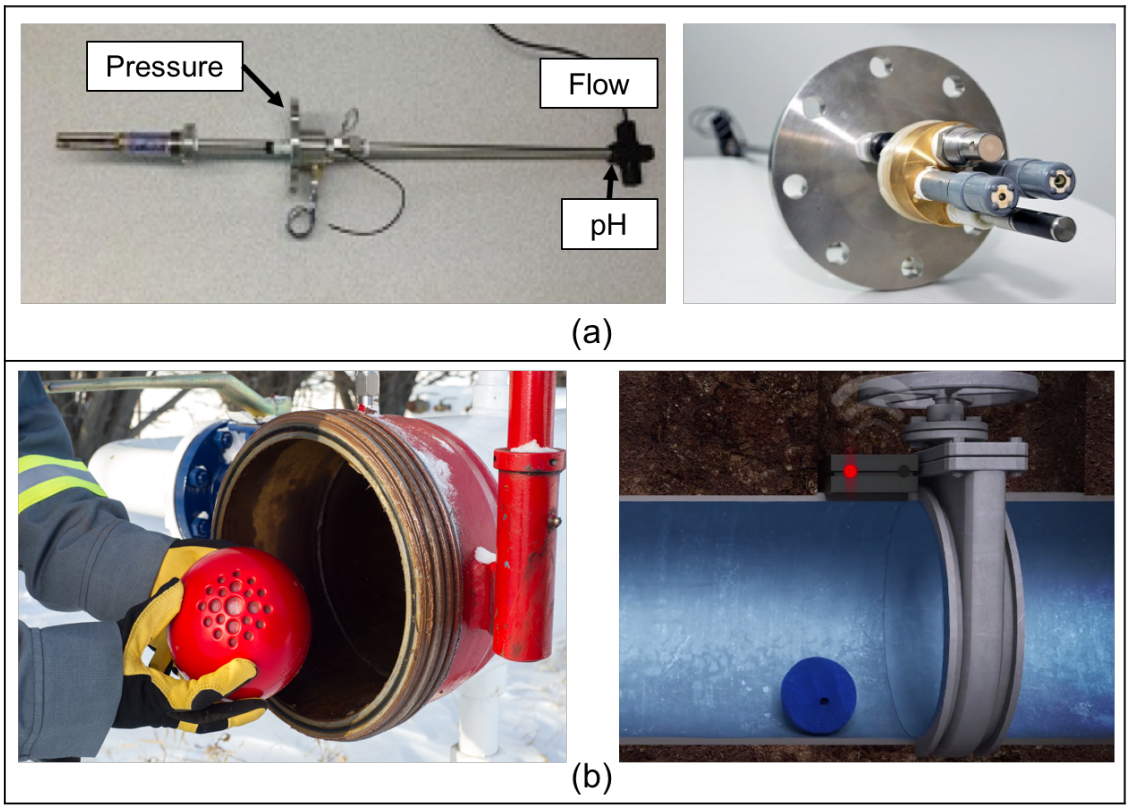


Figure 4.1: (a) WaterWiSe in-situ probe (b) SmartBall mobile deployment

2. Second, infrastructure events affecting the same community may have **have different severity** and hence cannot be treated uniformly. For example, simultaneous large pipe breaks would cause more damage to a community compared to a small pipe leak.

3. Third, diverse communities may be **vulnerable to different extents** to events based on their location, structure, demographics, built infrastructure, urbanization, etc.

In this chapter, we propose to leverage the advantages provided by static sensors (continuous monitoring, sensing range, accuracy) and those by mobile sensors (adaptive sensing, on-demand monitoring, low cost) to develop a hybrid (i.e. in-situ and mobile) sensor placement approach to provide adaptive monitoring of water networks. Our goal is to first plan and augment the placement of in-situ sensors with mobile sensor insertion infrastructure to quickly detect adverse events in the infrastructure and then determine locations from which to deploy mobile sensors to localize these events. Since the impact of events on the

community are tied to their severity and the time taken to localize them, our planning and deployment methodology needs to be able to quickly detect and localize high impact events.

There has been work on combining in-situ and mobile sensing in smart cities for pollution monitoring [95], community data collection [186], public safety [87] etc. However, achieving this in water networks presents several challenges. First, most of the network infrastructure is below ground, making it hard to deploy and operate sensors and involves other engineering efforts like developing mobile sensor insertion infrastructure. Second, the movement of mobile sensors is constrained by the direction and speed of the water flow that change over time. Finally, communication is a challenge in underground networks where wireless networks suffer from attenuation while wired approaches require much cost and effort.

Key Contributions of This Chapter:

- Methodology to model community vulnerability to determine the impact of events, that takes into account various geophysical, societal, demographical and topological factors (Section 4.2).
- Hybrid architecture that leverages the strengths of both in-situ and mobile sensors and combines it with the community geosocial factors to provide real-time adaptive monitoring of underground water distribution networks (Section 4.3).
- Approaches to model the components of the infrastructure, the occurrence and propagation of events, and their resulting impact on the community (Section 4.4).
- Algorithms to (a) perform network planning to determine the placement of static sensors and mobile sensor insertion infrastructure and (b) determine mobile sensor deployment locations with the goal of reducing costs and ensuring low community impact while maintaining event detection and localization accuracy (Section 4.5).
- Extensive evaluation of the performance of our proposed approach on three real-world

water networks from Maryland, Colorado, and Richmond, and comparisons with existing approaches in detecting dynamic events resulting in the loss of water quality/quantity (Section 4.6).

4.2 Community Impact of Events: A GeoSocial Approach

In this section, we further develop the notion of community impact due to adverse events in water networks. We argue that a comprehensive approach to sensor placement or deployment must consider a range of socioeconomic and geospatial factors and discuss how this differs from prior approaches to instrument water infrastructure. Specifically, we discuss the various geophysical, infrastructural, economic and societal factors that should be considered while modeling the impact of pipe failures or contamination events. We categorize these factors as follows:

- **Terrain and Topography:** Terrain is a major factor in determining the direction and speed of water flow. For instance, the flow of water would be faster down a slope than along a flat terrain. Also, the flow of water from events at a higher elevation can affect regions that are downstream. Hence, modeling the terrain elevation and gradient is important to determine the regions of the community that would get affected in the occurrence of a pipe failure or contamination event.

- **Event Characteristics and Network Structure:** The extent of impact of an event on the community is tightly coupled with the characteristics of the event. For instance, modeling the outflow of water when there is a pipe failure is essential to determine the extent of flooding of the community, while determining the levels of water consumption from different junctions would help quantify the impact if a contaminant was present.

The impact of an event would also depend on the topology of the water network (e.g.) a pipe failure upstream at a central distribution hub would have more impact on the community than a failure at an end node in the network. Similarly, modeling the network structure, presence of hydraulic infrastructure like pumps, and the pipe lengths are important to determine the concentration levels and the propagation of a contaminant through the water network.

- **Population Scale and Demographics:** The societal impact of a pipe break or contamination event is closely related to the scale of population that it affects. Adverse events in a population center would cause a larger disruption to the community than in a region of low population density. It is also important to understand the demographics of the region while modeling impact, since an event affecting an old-age home or a school could have a high adverse impact. Another factor is system redundancy. If no or little redundancy (alternative water supplies or conduits) exists, then the impact on the affected population would be higher.

- **Economic Impact:** The outflow of water from leaks and the consumption of contaminated water can also disrupt other lifelines and services and cause significant damage to property or the health of people in the community. Modeling the monetary costs associated with recovery and reconstruction activities can be used to model the impact of an event (e.g.) disruption of services can affect local businesses, thus causing secondary losses to the community.

- **Cascading Effects:** There is a potential for pipe breaks or contamination events to cascade. During pipe breaks, the seepage of water into nearby infrastructure can result in additional damage that could be exponentially more than the damage from flooding alone. An example of this type of cascading failure was observed during Hurricane Harvey this year[1] when a chemical plant in Crosby, Texas lost electrical power from backup generators because of flooding which eventually caused several explosions and

ensuing fires. Similarly, the consumption of contaminated water can result in the spread of severe health issues in the community, thereby potentially impacting the workforce resulting in reduced productivity for other industries.

In addition to modeling the geospatial aspects of failure, sensor placement methods must capture temporal metrics, (i.e.) detection time of events, while modeling impact and consequently response to failures. Our goal is to design sensor placement techniques that ensure adequate coverage of high impact regions (spatial aspect) with low detection times (temporal aspect). In the remainder of this chapter, we aim to answer the following questions:

- How do we define and quantify the spatio-temporal factors of impact accurately and meaningfully?
- How can we use these factors to model the vulnerability of a community?
- How do we use the notion of impact on a community to drive sensor placement in order to minimize the adverse impact of pipeline failures and contamination events?

4.3 Hybrid Adaptive Monitoring Architecture

In this section, we present our hybrid (in-situ plus mobile) architecture for the adaptive monitoring of water networks as shown in Figure 4.2. The physical infrastructure consists of the water distribution network, the surrounding community structure and the sensor deployments. In-situ or static sensors are installed on the pipes in contact with the water flow. Mobile sensors on the other hand, are inserted into, and extracted from the water flow through points that we denote as *Insertion/Extraction* (I/E) points. These could be manhole covers, fire hydrants or other specialized infrastructure. While the measurements

from static sensors are uploaded as continuous data streams, the data from mobile sensors are uploaded once they are extracted at an (I/E) point.

Our system architecture has two phases. The first phase involves network planning where we leverage network information, community structure (terrain, population, locations of key infrastructure), and event propagation models in order to model the impact of different events occurring at various locations in the network on the surrounding community. We then use these impact models to drive our planning algorithm to determine the locations for urban planners and water agencies to install static sensors and mobile sensor (I/E) points. The static sensors continuously monitor the water network and whenever they detect the occurrence of a pipe failure or contamination event, they determine a *region of interest*, which constitutes a subset of junctions and pipes where the sensors believe that an event has occurred. In the second phase, we determine the $Insertion/Extraction$ points from which to deploy mobile sensors to quickly cover the *region of interest* so as to ensure the minimal impact of events on the surrounding community.

The adaptive monitoring capabilities of our hybrid architecture results from (a) the ability to dynamically adjust the sensing resolution of different areas of the network on-demand through the deployment of mobile sensors, (b) being able to pick and choose the exact sensing capabilities (sensing rate, types of sensors) of the mobile sensors required to localize each event based on the input provided by the static sensor deployment, and (c) catering to the severity of different events by incorporating information from static sensors to determine the number of mobile sensors required to provide adequate coverage and measurements from the *region of interest*.

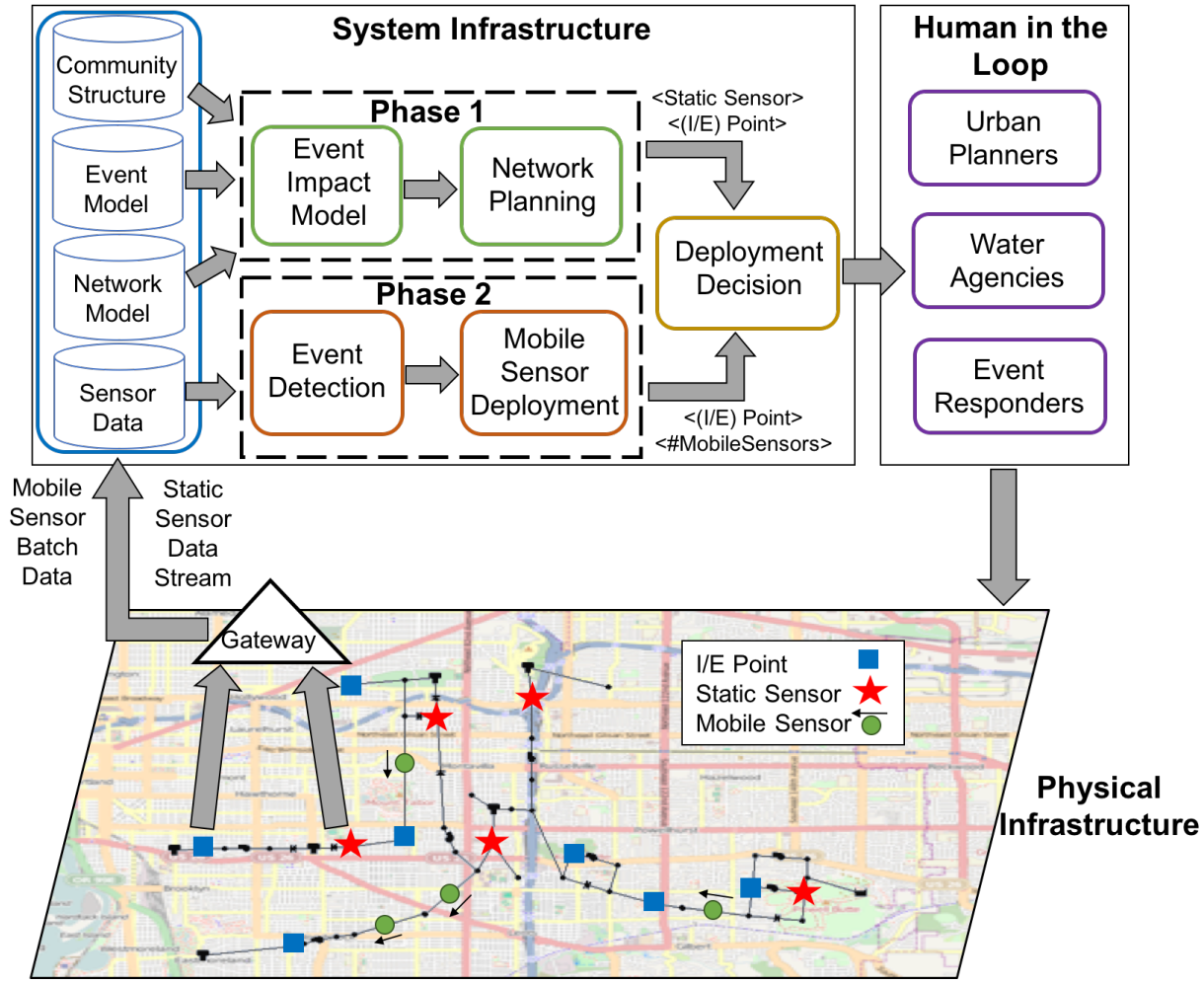


Figure 4.2: Hybrid Adaptive Monitoring Architecture

4.4 Modeling Impact in Water Infrastructure

In this section, we describe our methodology for modeling the various components of the proposed hybrid infrastructure, the occurrence and propagation of contamination and failure events, the community structure and the associated impact of these events.

4.4.1 Modeling Infrastructure Components

Since the propagation of different events is dependent on the network flow, an event may manifest itself at only a subset of junctions in the network. There is also a time delay associated with the manifestation that increases with distance from the event source. Therefore, it is important to model the water network and the sensing capabilities of both types of sensors to ensure the continuous monitoring of the network and the timely detection and localization of events. We use a hydraulic simulator EPANET [127] developed by the United States Environmental Protection Agency, which simulates the hydraulic behavior within pressurized water distribution pipe networks, to model the sensing capabilities of the sensors.

Modeling the Water Network: A water distribution network can be represented as a graph, where the vertices represent nodes and junctions, while the edges represent links (pipes, valves, and pumps). We denote the set of potential locations for the occurrence of event $\mathcal{E}$ in the network as $\mathcal{E} = \{e_1, e_2, ..., e_n\}$, where e_j refers to an event occurring at location j. We also define the set of potential static sensor locations and mobile sensor *Insertion/Extraction* points as $\mathcal{S}^{stat} = \{s_1^{stat}, s_2^{stat}, ..., s_n^{stat}\}$ and $\mathcal{S}^{mob} = \{s_1^{mob}, s_2^{mob}, ..., s_n^{mob}\}$ respectively, where s_i^{stat}, s_i^{mob} refer to a static sensor and an (I/E) point at location i respectively. There could be locations deep underground where installing in-situ sensors is not possible but could be reached by mobile sensors deployed from existing (I/E) infrastructure. Similarly, there could be places where installing specialized (I/E) infrastructure is infeasible due to network access or cost, where in-situ deployments are more useful. Figure 4.3 shows a sample water network with five nodes that we use as a running example to illustrate our modeling approach.

Modeling In-Situ Sensors: In our hybrid architecture, the static sensors are responsible for detecting the occurrence of any event and determine a *region of interest* by continuously monitoring the network. An event would cause static sensors to return measurements de-

(a) (b)

Figure 4.3: Running example of water network with its (a) elevation map and (b) key infrastructure information

$$\mathcal{M}_{dc} = \begin{array}{c} \\ s_1^{stat} \\ s_2^{stat} \\ s_3^{stat} \\ s_4^{stat} \\ s_5^{stat} \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array} \quad \mathcal{M}_{dt} = \text{(secs)} \begin{array}{c} \\ s_1^{stat} \\ s_2^{stat} \\ s_3^{stat} \\ s_4^{stat} \\ s_5^{stat} \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} 3 & 12 & \infty & 22 & \infty \\ 14 & 5 & \infty & \infty & \infty \\ 27 & 9 & 2 & \infty & \infty \\ 19 & \infty & \infty & 4 & \infty \\ \infty & 35 & 26 & \infty & 4 \end{pmatrix} \end{array}$$

(a)

$$\mathcal{M}_{tc} = \begin{array}{c} \\ s_1^{mob} \\ s_2^{mob} \\ s_3^{mob} \\ s_4^{mob} \\ s_5^{mob} \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} 1 & 7 & \infty & 6 & \infty \\ \infty & 1 & 3 & \infty & 4 \\ \infty & \infty & 1 & \infty & 1 \\ \infty & \infty & \infty & 1 & \infty \\ \infty & \infty & \infty & \infty & 1 \end{pmatrix} \end{array} \quad \mathcal{M}_{tt} = \text{(secs)} \begin{array}{c} \\ s_1^{mob} \\ s_2^{mob} \\ s_3^{mob} \\ s_4^{mob} \\ s_5^{mob} \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{pmatrix} 4 & 44 & \infty & 67 & \infty \\ \infty & 6 & 36 & \infty & 84 \\ \infty & \infty & 3 & \infty & 91 \\ \infty & \infty & \infty & 7 & \infty \\ \infty & \infty & \infty & \infty & 5 \end{pmatrix} \end{array}$$

(b)

$$\mathcal{M}_{fl} = \text{(ft.)} \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{array} \begin{array}{c} \begin{array}{cccc} \Delta_1 & \Delta_2 & \Delta_3 & \Delta_4 \end{array} \\ \begin{pmatrix} 3.4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6 \\ 0 & 0 & 2.7 & 0 \end{pmatrix} \end{array} \quad \mathcal{M}_{cl} = \text{(mg/L)} \begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{array} \begin{array}{c} \begin{array}{cccc} \Delta_1 & \Delta_2 & \Delta_3 & \Delta_4 \end{array} \\ \begin{pmatrix} 5.5 & 1.7 & 0 & 4.8 \\ 0 & 1.6 & 0.9 & 0 \\ 0 & 0 & 3.2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

(c)

Figure 4.4: Running example matrices for event detection and sensing capabilities of (a) static sensors and (b) mobile sensors, (c) Determining event propagation

pending on their ability to detect the event. The combination of these measurements can be used to identify junctions possibly affected by the event and hence track its propagation. Our goal is to then determine the potential event locations $e_j \in \mathcal{E}$ that can be detected by each static sensor location $s_i^{stat} \in \mathcal{S}^{stat}$ and the corresponding time taken to do so. To do this, we introduce an event at each potential location ($\mathcal{E}$) in EPANET. We then determine the sensor locations that can detect each event by monitoring the values of the requisite hydraulic variables for the event (pressure change for failures and contaminant concentration for contamination events). We build a *detection capability matrix* $\mathcal{M}_{dc}$, where the rows represent the potential static sensor locations and the columns represent the event locations.

The entries of the matrix are binary valued (0 or 1) depending on whether the static sensor is capable of detecting the event [114]. We denote the observed values of the hydraulic variable under normal conditions as v_i and as $\hat{v}_i$ once the event is introduced. We also set a detection threshold ϵ for each event. Then the values of $\mathcal{M}_{dc}$ for a particular event are computed as:

$$\mathcal{M}_{dc}[s_i^{stat}, e_j] = \begin{cases} 1, & \text{if } v_i - \hat{v}_i \geq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

We then store the corresponding time taken for the static sensors to detect an event in a *detection time matrix* $\mathcal{M}_{dt}$ as:

$$\mathcal{M}_{dt}[s_i^{stat}, e_j] = \begin{cases} \eta(s_i^{stat})_{e_j}, & \text{if } \mathcal{M}_{dc}[s_i^{stat}, e_j] = 1 \\ \infty, & \text{otherwise} \end{cases} \tag{4.2}$$

where $\eta(s_i^{stat})_{e_j}$ is the time taken for a static sensor s_i^{stat} to detect an event at e_j measured in seconds. The values of $\mathcal{M}_{dt}$ are set to infinity for the locations where a static sensor is incapable of detecting the event.

Example 4.1. *The detection capability and detection time matrices in Fig. 4.4(a) show the capability of static sensors to detect failures in the sample network. For instance, a static sensor at junction* 1 *can detect a leak at junction* 2 *with a delay of* 12 *seconds.*

Modeling Mobile Sensors: Once the static sensors determine a *region of interest*, the mobile sensors are then deployed in order to localize the event. Since the mobile sensors flow along with the water, at each junction that connects multiple pipes, a mobile sensor may flow into any one of the outlet pipes. Thus, sensors released at the same time and location can take a number of possible traversal paths. We account for this uncertainty by adopting a probabilistic approach to model the flow of mobile sensors in the network [143].

At each junction, we assign the probability of the sensor flowing through an outlet pipe connecting junction i and j as p_{ij}=f_{ij}/T_i, where f_{ij} is the flow rate through the outlet pipe and T_i is the total flow rate out of junction i. We maintain this junction-to-junction transition probability information in a matrix (M) which reflects the probability that a mobile sensor at junction i would reach junction j in a single step. This also implies that the probability of a mobile sensor from junction i reaching junction k after two steps can be computed as p_{ik}=p_{ij}+p_{jk} for all intermediate junctions j, which translates to computing M^2. We repeat this for n steps until there are no more transitions (i.e., all the probabilities are 0) and create a traversal probability matrix as:

$$\mathcal{T} = \sum_{k=1}^{n} M^k, \text{such that } M^k = \mathbf{0} \tag{4.3}$$

where $\mathbf{0}$ denotes the zero matrix and each entry of $\mathcal{T}$ denotes the probability of a mobile sensor traversing from one junction to another in any number of steps. Our goal is to then translate these probabilities into finding the number of mobile sensors required to traverse a junction with a minimum coverage probability p_c. This can be modeled as a binomial distribution $b(n, p)$, where p is the probability that a mobile sensor will traverse to a junction, $(1-p)$ the probability that it will not, and n the number of mobile sensors deployed. Hence, the probability that no mobile sensors will traverse to a junction can be represented as $(1-p)^n$. We define A as the event in which at least one mobile sensor traverses to a given junction. Our goal is to then determine n such that:

$$P(A) = P(1 - A') = 1 - (1-p)^n \geq p_c \tag{4.4}$$

where $P(A)$ is the probability of event A occurring (i.e.) at least one mobile sensor traverses to a given junction. From the above equation, we can obtain:

$$n \geq \left\lceil \frac{ln(1-p_c)}{ln(1-p)} \right\rceil \tag{4.5}$$

We see from Equation 4.5 that for low traversal probabilities, a larger number of mobile sensors need to be deployed. We then use Equation 4.5 to build a *traversal capability matrix* $\mathcal{M}_{tc}$ where the rows denote the potential mobile sensor (I/E) points ($\mathcal{S}^{mob}$) and the columns denote the event locations ($\mathcal{E}$). The entries of $\mathcal{M}_{tc}$ represent the minimum number of mobile sensors required to traverse to each event location with probability p_c where:

$$\mathcal{M}_{tc}[s_i^{mob}, e_j] = \begin{cases} n_{e_i e_j}, & \text{if } \mathcal{T}[s_i^{mob}, e_j] > 0 \\ \infty, & \text{otherwise} \end{cases} \tag{4.6}$$

We also build the corresponding traversal time matrix $\mathcal{M}_{tt}$ using the network flow rate information as the time taken for the mobile sensor to traverse from one junction to another:

$$\mathcal{M}_{tt}[s_i^{mob}, e_j] = \begin{cases} \theta(s_i^{mob}, e_j), & \text{if } \mathcal{M}_{tc}[s_i^{mob}, e_j] \neq \infty \\ \infty, & \text{otherwise} \end{cases} \tag{4.7}$$

where $\theta(s_i^{mob}, e_j)$ is the time taken for a mobile sensor to traverse from mobile sensor location s_i^{mob} to event location e_j denoted by the sum of the time taken to traverse each intermediate pipe which we compute using the flow rates and pipe lengths.

Example 4.2. *Figure 4.4(b) shows the traversal capability and traversal time matrices for mobile sensors in detecting contamination events in the sample network. We see that traversing from junction* 1 *to junction* 2 *requires at least* 7 *mobile sensors to achieve a* 95% *coverage probability and they take* 44 *seconds to reach junction* 2.

4.4.2 Modeling Events - Water Quality and Quantity

The occurrence of different events in the network can result in multiple hydraulic variables being affected. It is therefore important to accurately model the events to determine the manner of propagation of each event. Since different events can affect different parts of the community, we partition the community into smaller regions using a delaunay triangulator, Triangle [136]. Triangular grids allow for localized grid refinement and can easily conform to terrains with irregular shapes [12]. We denote the set of triangular regions as $\Delta = \{\triangle_1, \triangle_2, ..., \triangle_n\}$. We use EPANET to model the occurrence and propagation of contamination and failure events.

Contamination Events

In contamination events, a dissolved contaminant travels down the network with the same average velocity as the carrier fluid while at the same time reacting (either growing or decaying) at some given rate. Hence, contamination events can be detected by monitoring the chemical concentration levels in the water. We assume that at junctions, the mixing of fluid is complete and instantaneous.

Definition 4.1. *The propagation of contaminants in the water network can be formulated as:*

$$\frac{\partial C_i}{\partial t} = -u_i \frac{\partial C_i}{\partial x} + r(C_i) \tag{4.8}$$

where C_i is the concentration in pipe i as a function of distance x and time t, u_i is the flow velocity in pipe i and r is the rate of reaction as a function of concentration [127].

Contamination levels. The degree of contamination at each junction is an important factor to consider while measuring the impact of a contamination event on the community.

People consuming water from a contaminated junction within a region would be adversely affected. We measure this using EPANET by injecting a contaminant at each potential event location $e_j \in \mathcal{E}$ and simulating its spread throughout the network using Equation 4.8. We then build a *contaminant level matrix* $\mathcal{M}_{cl}$ where the rows and columns correspond to the contamination event locations ($\mathcal{E}$) and the triangular regions (Δ) respectively. We compute the entries of $\mathcal{M}_{cl}$ as:

$$\mathcal{M}_{cl}[e_j, \triangle_k] = \begin{cases} 0, & \triangle_k \text{ does not consume from } e_j \\ C(e_j)_{\triangle_k}, & \text{otherwise} \end{cases} \tag{4.9}$$

where $C(e_j)_{\triangle_k}$ denotes the concentration levels at $\triangle_k$ due to the contaminant intrusion at e_j.

Failure Events

Physical infrastructure failures, such as pipe leaks/breaks, cause a disturbance in the water flow resulting in a pressure wave that moves through the network with high velocity [98]. Past work has shown that the the velocity of water exiting from the leak orifice is faster than within the pipe causing a pressure drop, implying that pipe bursts can be identified by detecting changes in hydraulic pressure [113, 161].

Definition 4.2. *The outflow rate of water from a leak is defined as:*

$$Q = E_c \times p^{\beta} \tag{4.10}$$

where Q is the outflow rate from the leak, E_c is the effective leak area of the orifice, p is the pressure head at the leak and β is a constant [80].

Flood levels. One of the main factors influencing the impact of pipe failures on the com-

munity is the flooding resulting from water outflow and seepage from leaks. We capture this by simulating the outflow of water from a leak as well as its propagation along the surrounding terrain using a hydrodynamic flood simulation algorithm BreZo [130]. We simulate leak events in EPANET by introducing emitters. We then compute the outflow rate for each leak event $e_j \in \mathcal{E}$ using Equation (4.10) and provide this as input to the BreZo simulator in addition to the triangular regions (Δ) and the leak location (e_j). The BreZo simulator returns the regions affected by flooding and the corresponding flood levels. We use this information to build a *flood level matrix* $\mathcal{M}_{fl}$ consisting of the leak event locations as the rows and the triangular regions as the columns. The entries of $\mathcal{M}_{fl}$ are computed as:

$$\mathcal{M}_{fl}[e_j, \triangle_k] = \begin{cases} 0, & \text{leak at } e_j \text{ does not impact } \triangle_k \\ H(e_j)_{\triangle_k}, & \text{otherwise} \end{cases} \tag{4.11}$$

where $H(e_j)_{\triangle_k}$ denotes the maximum flood level at $\triangle_k$ due to a leak at e_j.

Example 4.3. *For the sample network, we measure the effects of failure and contamination events introduced at each junction for four triangular regions denoted by $\triangle$ in Figure 4.3(a). The resulting flood and contamination level matrices are shown in Figure 4.4(c).*

4.4.3 Modeling Community Structure and Event Impact

Estimating the impact of any event in the network on the community requires developing a model of the community structure. We then extract the following community information from each of the triangular community regions (Δ) :

1. **Critical Infrastructure:** We use mapping services to identify the presence of critical infrastructure such as healthcare, transportation, government facilities, education, etc within the boundaries of each region and assign relative importance scores to each of

these categories. We then compute the critical infrastructure score $\triangle_k^{inf}$, of region $\triangle_k$, as the sum of the scores of the infrastructure located in $\triangle_k$.

2. **Population Information:** We obtain the population density information of each of the triangular regions using census data and denote it as $\triangle_k^{pop}$.

3. **Elevation Information:** We build elevation maps for each of the triangular regions as the average elevation of its vertices and denote it as $\triangle_k^{ele}$.

4. **Demand:** We determine the average consumption of water by each triangular region $\triangle_k$ by averaging the total supply through each of the network junctions in $\triangle_k$, and denote this by $\triangle_k^{dem}$.

Example 4.4. *Figure 4.3(a) shows the triangular grids and the terrain information used to obtain $\triangle_k^{ele}$ while Figure 4.3(b) shows the presence of critical infrastructure around the water network to compute $\triangle_k^{inf}$.*

Measuring the impact of an event on the community ($\mathcal{I}^{\mathcal{E}}$) is dependent on the type of event that occurs in the network. Here, we present our methodology for measuring the impact of both pipe failures and contamination events.

Impact of Pipe Failures. A large pipe failure can cause flooding in the surrounding area, thus affecting the population present as well as the functioning of critical infrastructure. We thus compute the impact of a leak event at location j on region $\triangle_k$ as:

$$\mathcal{I}_{e_j}^{leak} = \mathcal{M}_{fl}[e_j^{leak}, \triangle_k] * (\triangle_k^{pop} + \triangle_k^{inf}) \tag{4.12}$$

where $\mathcal{M}_{fl}[e_j^{leak}, \triangle_k]$ is the level of flooding caused by the leak event, $\triangle_k^{pop}$ is the population density, and $\triangle_k^{inf}$ is the critical infrastructure score of the region.

Impact of Contamination Events. The impact of a contamination event would depend on the amount of contaminated water consumed as well as the number of people consuming

the water. We estimate the impact of a contamination event at location j on region $\triangle_k$ as:

$$\mathcal{I}_{e_j}^{cont} = \mathcal{M}_{cl}[e_j^{cont}, \triangle_k] * (\triangle_k^{pop} + \triangle_k^{dem}) \tag{4.13}$$

where $\mathcal{M}_{cl}[e_j^{leak}, \triangle_k]$ is the contamination levels caused by the event, $\triangle_k^{pop}$ is the population density, and $\triangle_k^{dem}$ is the consumption demand of the region.

4.5 Network Planning and Deployment Algorithms

As described in Section 4.3, given a water network, our proposed architecture determines the placement of static sensors and *Insertion/Extraction* points to quickly identify the *regions of interest* and then identifies the deployment locations of mobile sensors to provide rapid localization of events. In this section, we present our network planning and deployment algorithms. The goal of our planning algorithm is to simultaneously determine locations to place both static sensors and mobile sensor *Insertion/Extraction* points while providing high utility and incurring low costs. We define a sensor placement $\mathcal{P}$ to consist of a set of static sensor locations and mobile sensor (I/E) points (i.e.) $\mathcal{P} \subseteq (\mathcal{S}^{stat} \bigcup \mathcal{S}^{mob})$. The utility of a placement can be measured in terms of the impact caused by events in the network due to the delay in their detection and localization. A sensor placement that provides high utility is thus one that 1) quickly detects and localizes high impact events, 2) results in low overall impact on the community, and 3) incurs low costs.

We denote $\mathcal{C}_i^{stat}, \mathcal{C}_i^{mob}$ as the set of event locations $(\mathcal{E})$ that can be detected by a static sensor placed at, or traversed to by mobile sensors deployed from location i respectively. We also define $cost_s : cost_m$ as the cost ratio of static and mobile sensors and $\mathcal{C}^{cov}$ as the set of event locations detectable by the deployment. We acknowledge that modeling the various costs involved is complex and dependent on the event type and severity, and use sensor cost ratios to determine the relative utility provided by in-situ and mobile sensors. The objective of

our planning algorithm is to then identify a minimum cost placement set $\mathcal{P} = (\mathcal{P}^{stat} \cup \mathcal{P}^{mob})$ that covers the set of all detectable events such that its overall utility is maximized. This is equivalent to the *weighted set cover* problem.

Definition 4.3. *(Weighted Set Cover) Let $\mathcal{L}$ be a finite set of elements and $\mathcal{C}$=$\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n\}$ be a set of subsets of $\mathcal{L}$ with weights $\mathcal{W} = \{w_1, w_2, ..., w_n\}$. The goal is to find a set $\mathcal{C}_s \subseteq \mathcal{C}$ such that all the elements are covered by $\mathcal{C}_s$ (i.e.) $\bigcup C_k = \bigcup C_i, \forall C_k \in \mathcal{C}_s, \forall C_j \in \mathcal{C}$, and the sum of weights in $\mathcal{C}_s$ is minimized (i.e.)* $\min(\sum w_k, \forall C_k \in \mathcal{C}_s)$.

Using this definition, if $\mathcal{C}$ is the collection of event locations covered by each sensor, and $\mathcal{W}$ the corresponding costs of static and mobile sensors, finding a set cover $\mathcal{C}_s$ is a solution to the *WSC* problem and hence shows the equivalence. The weighted set cover problem is NP-hard [75] thus implying that finding a solution for the planning problem is also computationally complex. We present our approximate Hybrid Impact Driven (HID) placement algorithm as described in Algorithm 1:

- For a given budget $\mathcal{B}$ and set of event locations $\mathcal{E}$, we iterate over the set of potential sensor locations and determine the utility of placing a static sensor or an *Insertion/Extraction* point at each location.
- For a static sensor s_i^{stat}, we compute its utility for event e_j as a function of the impact caused in the time taken for the sensor to detect the event:

$$U_{e_j}^{stat}(s_i^{stat}) = \mathcal{I}_{e_j} / \mathcal{M}_{dt}[s_i^{stat}, e_j] \tag{4.14}$$

- The utility for an (I/E) point s_i^{mob} depends on the impact caused in the time taken for the existing static sensors to determine the *region of interest* and for the mobile sensors to traverse to the event location:

$$U_{e_j}^{mob}(s_i^{mob}) = \mathcal{I}_{e_j} / (\delta_{dt}[e_j] + \mathcal{M}_{tt}[s_i^{mob}, e_j]) \tag{4.15}$$

where $\delta_{dt}[e_j]$ is the shortest time taken for the placed static sensors to detect e_j.

- We then compute the total utility as a function of the cost incurred to achieve the above utilities. This is

$$\mathcal{U}^{stat}(s_i^{stat}) = \sum_{j=1}^{|\mathcal{E}|} U_{e_j}^{stat}(s_i^{stat})/cost_s \tag{4.16}$$

for static sensors, and

$$\mathcal{U}^{mob}(s_i^{mob}) = \sum_{j=1}^{|\mathcal{E}|} U_{e_j}^{mob}(s_i^{mob})/(\mathcal{M}_{tc}[s_i^{mob}, e_j] * cost_m) \tag{4.17}$$

for mobile sensor *Insertion/Extraction* points, where $\mathcal{M}_{tc}[s_i^{mob}, e_j]$ is the minimum number of mobile sensors needed to be deployed from s_i^{mob}.

- At the end of each iteration, we choose the sensor type and location pairing with the largest utility and add it to the placement set. We do this until either the budget is exhausted or the network has been covered.

The algorithm is guaranteed to complete since every detectable event location has at least one static sensor location that covers it. The worst case running time of the algorithm, $\mathcal{O}((|\mathcal{S}^{stat}|+|\mathcal{S}^{mob}|)|\mathcal{E}|^2)$, occurs when at each iteration, only one new event location is covered. This would result in long runtimes for large scale water networks. We however use the concept of *submodularity* to significantly reduce the number of utility computations in each iteration, thus reducing the runtime [102].

Definition 4.4. *(Submodularity) Let $\mathcal{C}$ be a finite set and f be a set function. For all subsets $\mathcal{C}_s \subseteq \mathcal{C}_r \subseteq \mathcal{C}$, and elements $c_i \in \mathcal{C} \setminus \mathcal{C}_r$, f is submodular whenever $f(\mathcal{C}_s \cup c_i) - f(\mathcal{C}_s) \geq f(\mathcal{C}_r \cup c_i) - f(\mathcal{C}_r)$.*

Theorem 4.1. *The utility functions $\mathcal{U}^{stat}$, $\mathcal{U}^{mob}$ are submodular.*

Algorithm 1 HID Placement Algorithm

Input: $\mathcal{S}^{stat}, \mathcal{S}^{mob}, \mathcal{E}, \mathcal{M}_{dc}, \mathcal{M}_{dt}, \mathcal{M}_{tc}, \mathcal{M}_{tt}, \mathcal{B}, cost_s, cost_m,$
$\mathcal{P}^{stat}, \mathcal{P}^{mob}, \mathcal{C}^{stat}, \mathcal{C}^{mob}, \mathcal{C}^{cov}$
Output: $\mathcal{P} = (\mathcal{P}^{stat} \cup \mathcal{P}^{mob})$
Initial Conditions: $\mathcal{P}^{stat} = \emptyset, \mathcal{P}^{mob} = \emptyset, \mathcal{C}^{cov} = \emptyset,$
while $C^{cov} \neq |\mathcal{E}|$ **and** $\mathcal{B} > 0$ **do**
  for $i = 1 \rightarrow |\mathcal{S}^{stat} \cup \mathcal{S}^{mob}|$ **do**
    if $s_i^{stat} \notin \mathcal{P}^{stat}$ **then**
      for $j = 1 \rightarrow |\mathcal{E}|$ **do**
        $U_{e_j}^{stat}(s_i^{stat}) = \mathcal{I}_{e_j} / \mathcal{M}_{dt}[s_i^{stat}, e_j]$
      $\mathcal{U}^{stat}(s_i^{stat}) = \sum_{j=1}^{|\mathcal{E}|} U_{e_j}^{stat}(s_i^{stat}) / cost_s$
    if $s_i^{mob} \notin \mathcal{P}^{mob}$ **then**
      for $j = 1 \rightarrow |\mathcal{E}|$ **do**
        $U_{e_j}^{mob}(s_i^{mob}) = \mathcal{I}_{e_j} / (\delta_{dt}[e_j] + \mathcal{M}_{tt}[s_i^{mob}, e_j])$
      $\mathcal{U}^{mob}(s_i^{mob}) = \sum_{j=1}^{|\mathcal{E}|} U_{e_j}^{mob}(s_i^{mob}) / (\mathcal{M}_{tc}[s_i^{mob}, e_j] * cost_m)$
    $\mathcal{U}(s_i) = max(\mathcal{U}^{stat}(s_i^{stat}), \mathcal{U}^{mob}(s_i^{mob}))$
  $\mathcal{U}^{max}(s_v) = \{\mathcal{U}(s_i) : max(\mathcal{U}(s_i)), \forall i : 1 \rightarrow |\mathcal{S}|\}$
  if $\mathcal{U}^{max}(s_v) = \mathcal{U}^{stat}(s_v)$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} - cost_s$
    $\mathcal{C}^{cov} \leftarrow \mathcal{C}^{cov} \cup \mathcal{C}_v^{stat}$
    $\mathcal{P}^{stat} \leftarrow \mathcal{P}^{stat} \cup s_v$
  else
    $\mathcal{B} \leftarrow \mathcal{B} - (\mathcal{M}_{tc}[s_v^{mob}, \mathcal{E}] * cost_m)$
    $\mathcal{P}^{mob} \leftarrow \mathcal{P}^{mob} \cup s_v$

Proof. We see from the formulation of the event utility functions $\mathcal{U}^{stat}, \mathcal{U}^{mob}$ that they depend on the impact caused by the event and the time taken to detect and localize it. Since, the impact formulation derived from Section 4.4.3 is independent of the sensor deployment, the submodularity of the event utility functions depends on the detection time $\mathcal{M}_{dt}$ and traversal time $\mathcal{M}_{tt}$.

For the static event utility function $\mathcal{U}^{stat}$, consider two placement sets $\mathcal{P}^{stat} \subseteq \mathcal{Q}^{stat} \subseteq \mathcal{S}^{stat}$. Given an event $e_j \in \mathcal{E}$ that can be detected by a static sensor installed at location i such that $s_i^{stat} \in \mathcal{S}^{stat} \setminus \mathcal{P}^{stat}$. Depending on the time taken for s_i^{stat} to detect e_j, there are three cases:

Case 1

$$\mathcal{M}_{dt}[s_i^{stat}, e_j] \geq \min(\mathcal{M}_{dt}[\mathcal{P}^{stat}, e_j]) \tag{4.18}$$

This implies,

$$\min(\mathcal{M}_{dt}[\mathcal{P}^{stat} \cup \{s_i\}, e_j]) = \min(\mathcal{M}_{dt}[\mathcal{P}^{stat}, e_j]) \tag{4.19}$$

$$\min(\mathcal{M}_{dt}[\mathcal{Q}^{stat} \cup \{s_i\}, e_j]) = \min(\mathcal{M}_{dt}[\mathcal{Q}^{stat}, e_j]) \tag{4.20}$$

And hence,

$$\mathcal{U}^{stat}(\mathcal{P}^{stat} \cup \{s_i^{stat}\}) - \mathcal{U}^{stat}(\mathcal{P}^{stat}) = \mathcal{U}^{stat}(\mathcal{Q}^{stat} \cup \{s_i^{stat}\}) - \mathcal{U}^{stat}(\mathcal{Q}^{stat}) = 0 \tag{4.21}$$

Case 2

$$\min(\mathcal{M}_{dt}[\mathcal{Q}^{stat}, e_j]) \leq \mathcal{M}_{dt}[s_i^{stat}, e_j] < \min(\mathcal{M}_{dt}[\mathcal{P}^{stat}, e_j]) \tag{4.22}$$

This implies,

$$\mathcal{U}^{stat}(\mathcal{Q}^{stat} \cup \{s_i^{stat}\}) = \mathcal{U}^{stat}(\mathcal{Q}^{stat}) \tag{4.23}$$

And hence,

$$\mathcal{U}^{stat}(\mathcal{P}^{stat} \cup \{s_i^{stat}\}) - \mathcal{U}^{stat}(\mathcal{P}^{stat}) \geq \mathcal{U}^{stat}(\mathcal{Q}^{stat} \cup \{s_i^{stat}\}) - \mathcal{U}^{stat}(\mathcal{Q}^{stat}) \tag{4.24}$$

Case 3

$$\mathcal{M}_{dt}[s_i^{stat}, e_j] < \min(\mathcal{M}_{dt}[\mathcal{Q}^{stat}, e_j]) \tag{4.25}$$

Here,

$$\mathcal{U}^{stat}(\mathcal{P}^{stat} \cup \{s_i^{stat}\}) \geq \mathcal{U}^{stat}(\mathcal{Q}^{stat} \cup \{s_i^{stat}\}) \tag{4.26}$$

$$\mathcal{U}^{stat}(\mathcal{P}^{stat}) \leq \mathcal{U}^{stat}(\mathcal{Q}^{stat}) \tag{4.27}$$

due to the non-decreasing property of $\mathcal{U}^{stat}$. Hence, we get,

$$\mathcal{U}^{stat}(\mathcal{P}^{stat} \cup \{s_i^{stat}\}) - \mathcal{U}^{stat}(\mathcal{P}^{stat}) \geq \mathcal{U}^{stat}(\mathcal{Q}^{stat} \cup \{s_i^{stat}\}) - \mathcal{U}^{stat}(\mathcal{Q}^{stat}) \tag{4.28}$$

From Equations 4.21, 4.24, and 4.28 we prove that $\mathcal{U}^{stat}$ is submodular. Similarly we can prove that $\mathcal{U}^{mob}$ is also submodular.

□

Using this greedy approach in the HID placement algorithm gives an approximation ratio of $(1-1/e)$ similar to the weighted set cover problem [102]. Hence, in a given iteration of the algorithm if $\mathcal{U}(s_1) \geq \mathcal{U}(s_2) \geq \mathcal{U}(s_3) \geq ..., \geq \mathcal{U}(s_n)$, then s_1 would be added to the placement. Then in the next iteration, if $\mathcal{U}(s_2) \geq \mathcal{U}(s_3)$, we can conclude that $\mathcal{U}(s_2) \geq \mathcal{U}(s_i), \forall i \geq 3$, thus reducing the number of evaluations needed at each step of the algorithm.

Once the locations of static sensors and mobile sensor *Insertion/Extraction* points have been determined, we need to identify the mobile sensor deployment locations. This depends on the direction and velocity of water flow in the network as well as the junctions that need to be localized. Given the junctions within the *regions of interest* $\mathcal{R}$, our objective is to identify the subset of (I/E) points determined by our planning algorithm at which mobile sensors need to be deployed (i.e.) $\mathcal{D} \subseteq \mathcal{P}^{mob}$. We summarize our deployment algorithm as shown in Algorithm 2 below:

- For every junction e_j in the *region of interest* $\mathcal{R}$, we identify the (I/E) points in

Algorithm 2 Mobile Sensor Deployment Algorithm

Input: $\mathcal{R}, \mathcal{M}_{tt}, \mathcal{M}_{tc}, \mathcal{P}^{mob}$
Output: $\mathcal{D} \subseteq \mathcal{P}^{mob}$
for all $e_j \in \mathcal{R}$ **do**
    $d = \emptyset, d^t = \emptyset$
    for $i = 1 \rightarrow |\mathcal{P}^{mob}|$ **do**
        if $\mathcal{M}_{tc}[\mathcal{P}_i^{mob}, e_j] > 0$ **and** $\mathcal{M}_{tt}[\mathcal{P}_i^{mob}, e_j] < d^t$ **then**
            $d^t = \mathcal{M}_{tt}[\mathcal{P}_i^{mob}, e_j]$
            $d = \mathcal{P}_j^{mob}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup d$

the placement set $\mathcal{P}^{mob}$, from which mobile sensors can traverse to e_j (i.e.) where $\mathcal{M}_{tc}[\mathcal{P}_i^{mob}, e_j] > 0$.

- We then determine the (I/E) point with the shortest traversal time and add it to the deployment set $\mathcal{D}$.

- We repeat this till the entire *region of interest* has been localized.

More complex models for mobile sensor deployment are possible that exploit existing control systems in the network such as pumps and valves to change the direction and speed of water flow that could result in fewer (I/E) points needed to ensure the reachability and coverage of mobile sensors. However, this requires detecting the state of the systems and estimating their levels of functionality in the aftermath of the event. Our proposed approach, though more conservative, results in deployment solutions unaffected by effects of the event on these systems.

4.6 Experiments

In this section, we evaluate our proposed hybrid adaptive monitoring architecture for both failure and contamination events. We compare the performance of our approach to existing sensor deployment approaches using water networks of varying scale and validate our

Network	Length (km)	Demand $\times 10^3$ (m^3/day)	#Pipes	#Junctions
WSSC	32.46	1.57	316	299
Richmond	75.61	15.12	948	865
WCR	383.59	82.95	1985	1782

Table 4.1: Summary of real-world water distribution network infrastructure

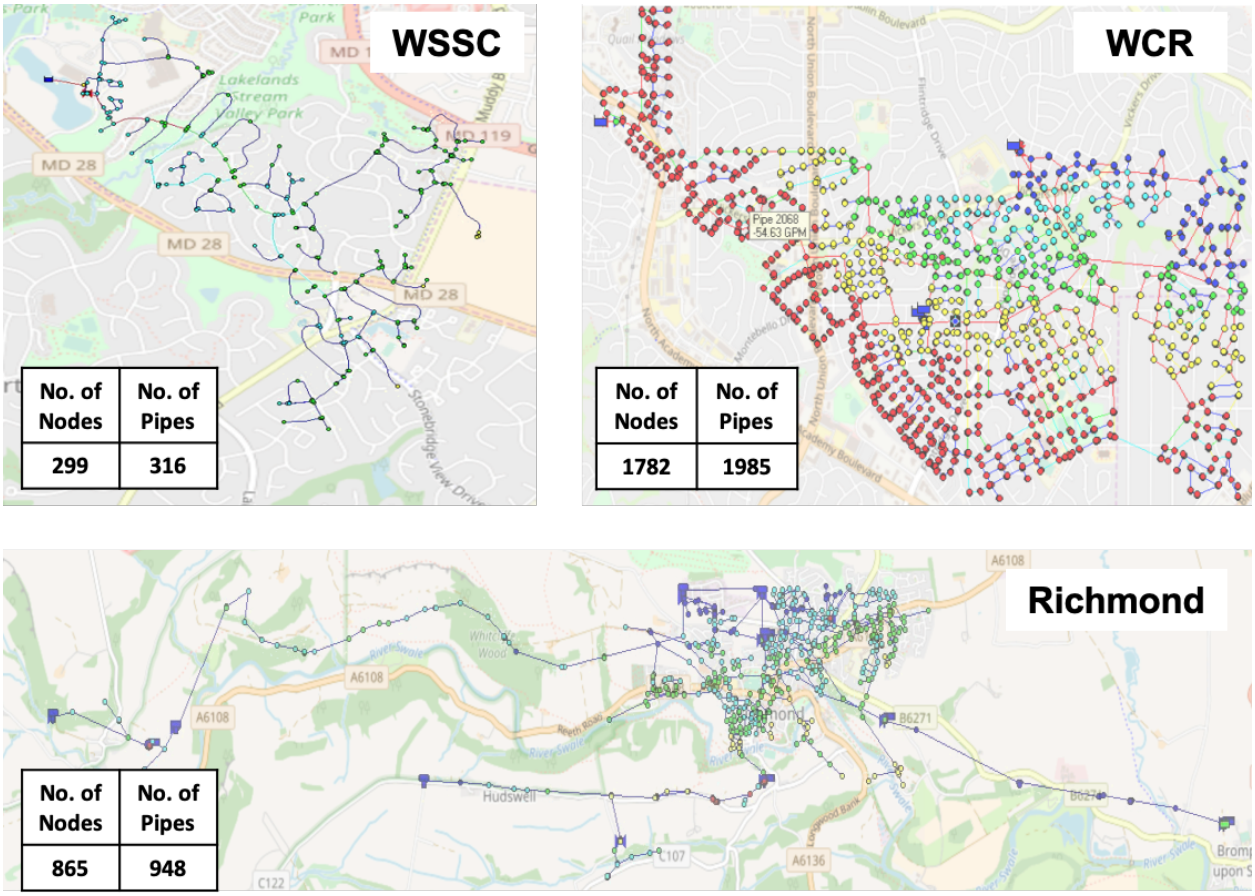


Figure 4.5: Infrastructure network layouts of WSSC, WCR and Richmond

approach under multiple event scenarios.

4.6.1 Experimental Setup

Water Networks. We evaluate our proposed architecture using three real-world water networks of varying scale - (1) a subzone of the Washington Suburban Sanitary Commis-

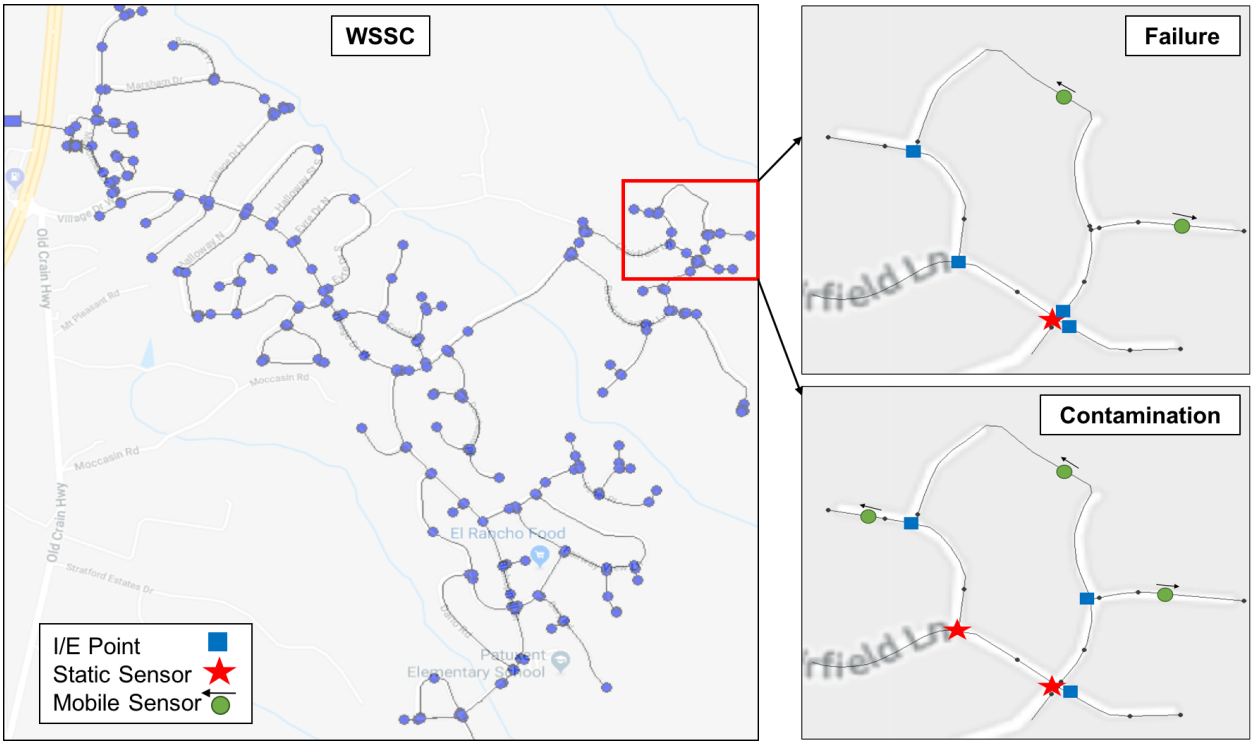


Figure 4.6: WSSC network and subset of sensor deployment for failure and contamination events

sion's (WSSC) water service area in Montgomery County, Maryland, (2) a model for the Wolf-Cordera Ranch (WCR) in Colorado Springs, Colorado, and (3) the Richmond water distribution system, part of the Yorkshire water supply area in the U.K. The data for (1) was obtained from WSSC while for (2) and (3) from [105]. Figure 4.5 shows the layout of the water distribution infrastructure for all three networks, and a detailed summary of the network structure is shown in Table 4.1. In addition, Figure 4.6 shows the layout of the WSSC network and its surrounding community and shows the placement of in-situ sensors and mobile sensor (I/E) points determined by our proposed HID placement algorithm for a subset of the network.

Comparison Approaches. We compare our proposed Hybrid Impact Driven planning and deployment approach *(HID)* [165] to three existing sensor placement approaches in water networks - Static Coverage Driven *(SCD)* [113], Hybrid Coverage Driven *(HCD)* [112], and Static Impact Driven *(SID)* [161]. Here, static and hybrid refer to the sensor types being

Approach	Failure (in-situ/mobile)			Contamination (in-situ/mobile)		
	WSSC	Richmond	WCR	WSSC	Richmond	WCR
SCD	61/0	241/0	402/0	83/0	270/0	430/0
SID	89/0	310/0	489/0	110/0	367/0	512/0
HCD	11/54	104/126	147/261	41/54	141/144	154/288
HID	**20/62**	**89/240**	**122/343**	**27/73**	**112/261**	**186/313**

Table 4.2: Number of sensors deployed (in-situ/mobile) by each approach for failure and contamination events.

used in the approach. The *SCD* approach iteratively selects static sensor locations based on the number of event locations covered and their ability to distinguish between pairs of events. The *HCD* approach uses a cross entropy based methodology to select a percentage of junctions to install static sensors followed by determining mobile sensor release points. The *SID* approach iteratively determines locations to install static sensors based on their achieved impact mitigation of the event on the community

Table 4.2 shows the number of in-situ sensors and mobile sensors in the deployments determined by each approach. We observe that sensor deployments for pipe failure detection use fewer sensors to detect and localize failure events since the event propagation (movement of pressure wave) is faster as compared to the contaminant flow which is restricted by the flow velocity of the water in the network. We also see that the hybrid approaches result in the deployment of far fewer in-situ sensors thus reducing the costs incurred. However, the deployments resulting from impact driven approaches use more sensors than their coverage driven counterparts since they attempt to quickly localize high impact events resulting in more sensors being deployed to cover vulnerable regions.

Event Scenarios. We compare the performance of the sensor deployments resulting from each of the approaches for the following event scenarios.

1. *Geo-correlated events:* We simulate the cascading effects of failure and contamination events by introducing them in spatially clustered locations ranging from 5% to 50% of

the network's junctions where the number of events in each cluster is uniform.

2. *Critical events:* We then introduce failure and contamination events in the top 5% to 30% junctions ordered by impact.

4.6.2 Evaluating Effectiveness of Hybrid Sensor Deployments

We compare the effectiveness of the sensor deployments resulting from each approach using three metrics - (a) Detection and localization times, (b) Impact caused, (c) Cost effectiveness. In order to determine the impact of failure and contamination events as described in Section 4.4.3, we obtain community structure information by building the terrain elevation map ($\triangle^{ele}$) using elevation data from [94], obtain population density information ($\triangle^{pop}$) from census data [23], mine the coordinates of critical infrastructure ($\triangle^{inf}$) in the area using the OpenStreetMap service [55], and determine the demand of water at each junction ($\triangle^{dem}$) from the network model.

Evaluating Detection and Localization Times: For each introduced failure or contamination event in the above event scenarios, we determine the shortest time taken to detect and localize the event by the sensors deployed by each approach. We then compare the average of these shortest times for all the introduced events. Figure 4.9(a) shows the comparison of the average detection times by each of the approaches for geo-correlated failure and contamination events. We see that in general, the detection and localization of failure events ($98 - 398$ sec) is faster than contamination events ($1010 - 7035$ sec). Also, the detection and localization time of contamination events increases with the size of the network, while that of failure events remains consistent. This happens because the high speed pressure wave resulting from pipe failures can be detected quickly even in larger networks as compared to the slower moving contaminant flow. We observe that the coverage driven approaches (*SCD*,*HCD*) take a longer time on average to detect and localize events as com-

pared to impact driven approaches (*SID*,*HID*) since their coverage based objective results in sparser sensor deployments. We also observe that the proposed hybrid *HID* approach takes approximately 9% longer to detect and localize events as compared to the in-situ based *SID* approach.

Extent of Impact Caused: We then compare the approaches based on the impact caused to the community by the failure and contamination event scenarios before their deployments can detect and localize the events. For each introduced event, we determine its impact (Section 4.4.3) caused as a function of the shortest time taken for each approach's sensor deployment to detect and localize it. We then compute the average normalized impact caused over all the introduced events (Figure 4.9(b)). We see that for geo-correlated events, the impact driven approaches (*SID*,*HID*) result in much lower impacts on average since they prioritize the quick detection of high impact events and the coverage of critical regions. We observe that the proposed *HID* approach results in upto nearly 30% lesser impact than the *SID* approach due to the faster localization of events using mobile sensors and upto nearly 79% lesser impact than the coverage based approaches.

Sensor type cost ratio: Here, we determine the influence of the cost ratio between mobile and static sensors on their proportion in the sensor deployment resulting from the proposed *HID* approach by varying the cost ratio of mobile to static sensors from 1:1 to 1:10. Figure 4.7 shows the proportion of static and mobile sensors for the WSSC network for (a) failure and (b) contamination events. We see that there is a stabilization in the proportions at a 1:5 cost ratio for the WSSC network. We observe that this ratio increases with an increase in the size of the network.

Comparison of coverage and cost of deployment: We then use the 1:5 cost ratio to compare the costs of the deployments resulting from each of the four approaches. We vary the number of sensors from 10% to 100% of the total number of sensors in each deployment and compare the costs incurred and the coverage of the network achieved at each step for

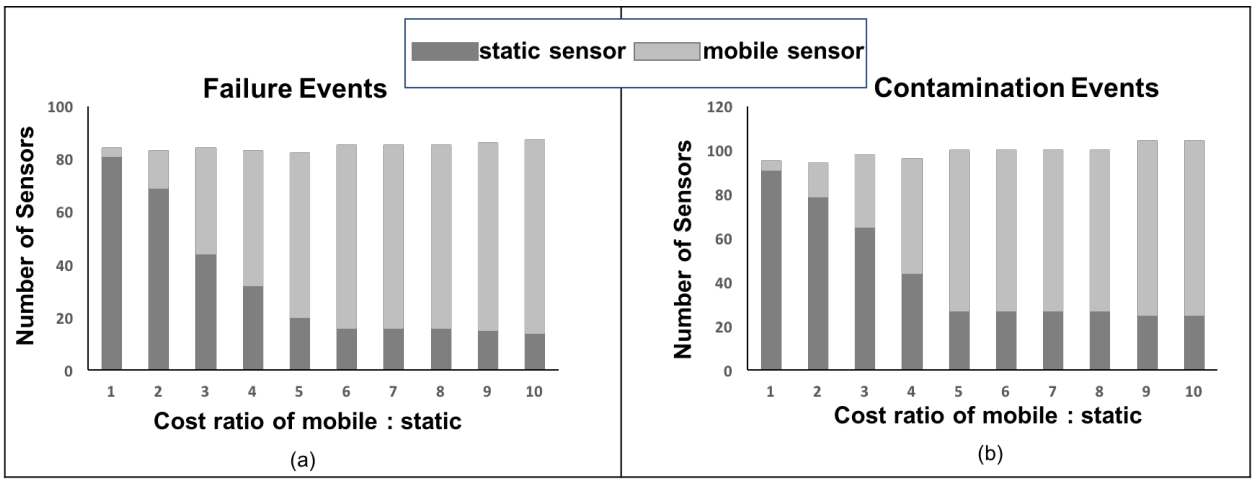


Figure 4.7: Proportion of in-situ and mobile sensors in proposed hybrid approach deployment with varying mobile:static sensor cost ratios for the WSSC network

the WSSC network. Figure 4.8 shows that approaches using only in-situ sensors (*SID*,*SCD*) incur much larger costs than the hybrid approaches (*HID*,*HCD*). We also observe that the coverage driven approaches achieve higher network coverage using lesser number of sensors.

Cost effectiveness of deployments: We evaluate the normalized cost effectiveness of the deployments as a function of the total impact caused and the total cost incurred. We observe from Figure 4.9(c) that for geo-correlated events, the proposed *HID* approach proves to be the most cost effective by upto nearly 52% over the *SID* approach and upto nearly 68% over the coverage based approaches. This would improve as the cost ratio between mobile and static sensors increases.

Performance under Critical Events: Due to space constraints, we present the results of the comparison of the four approaches for the critical events scenario in the appendix. We observe that the hybrid driven approaches detect critical events much faster than coverage based approaches and result in lower impacts of events. We also see that the proposed *HID* approach remains the most cost effective by upto nearly 40% over the *SID* approach.

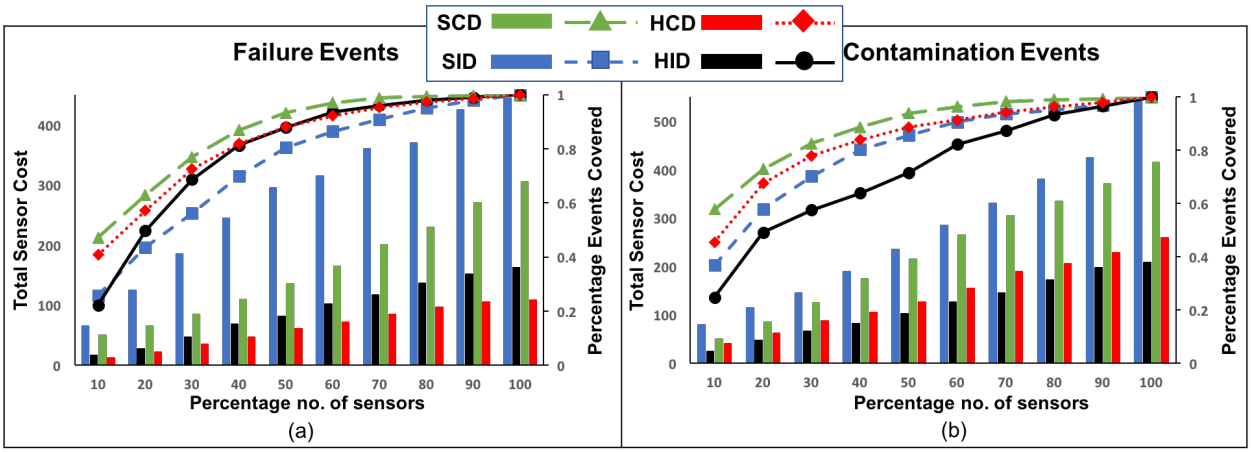
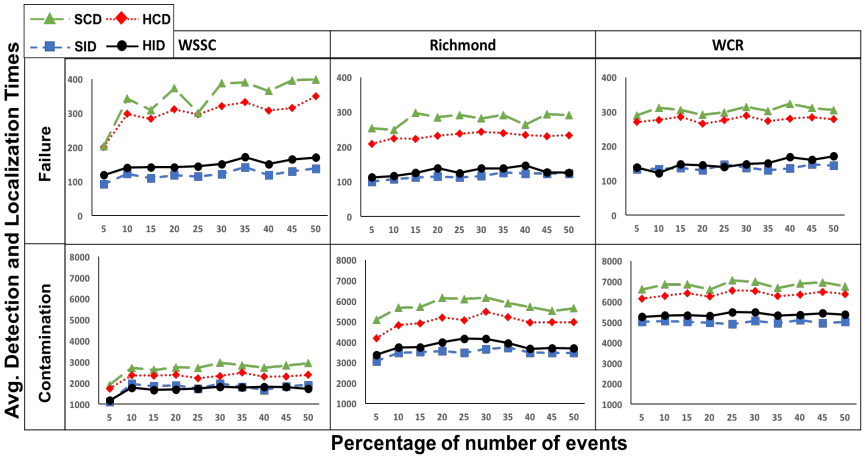


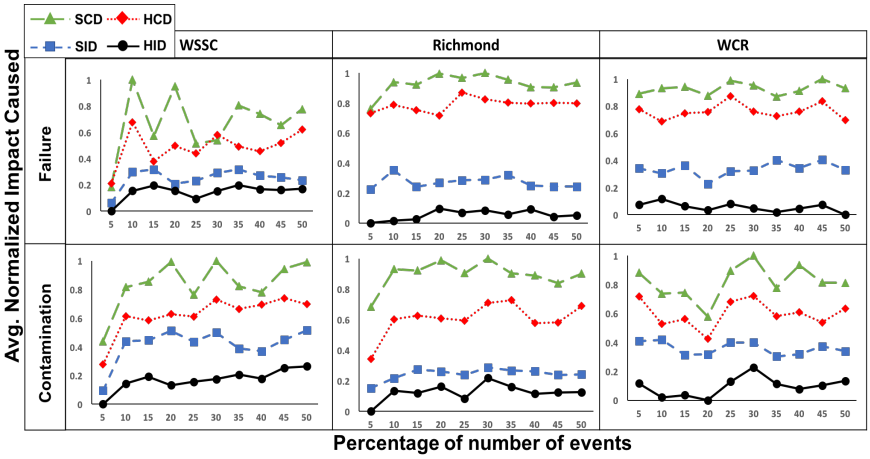
Figure 4.8: Progression of sensor deployment costs and achieved event coverage with increasing number of sensors by the approaches for the WSSC network

4.7 Chapter Summary and Discussion

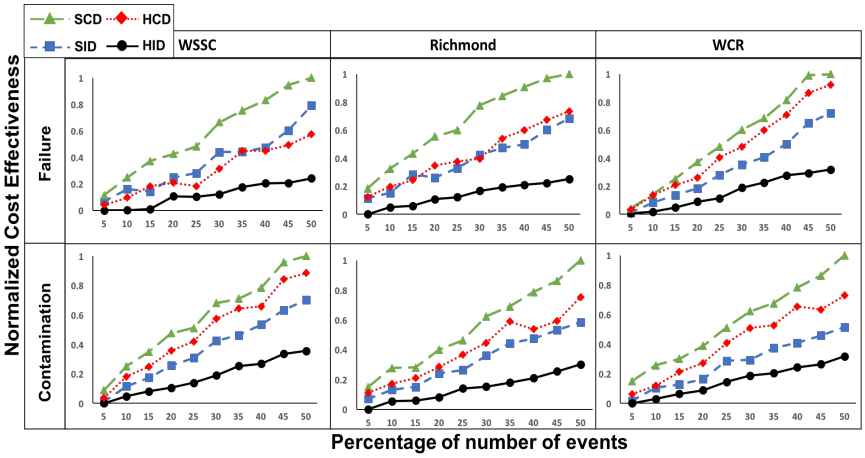
In this chapter, we present a novel approach to sensor placement for event detection in community infrastructure with a focus on water distribution networks. Our approach is based on the notion of impact of events on the community. We presented a methodology to characterize and quantify the various geosocial factors influencing the impact of an event, and to incorporate this definition of impact into the sensor placement decision making process. We developed an impact-driven architecture for sensor placement to quickly detect and localize adverse events like pipe breaks and contamination. Our proposed architecture leverages in-situ and mobile sensing to provide a cost-effective solution that minimizes the impact of infrastructure events on the community. We presented a two-phase approach that incorporates information about the network, impact of events, and the community to determine locations to install sensing infrastructure and to deploy mobile sensors and evaluate its effectiveness using real-world water networks of varying scale. Our results show that this hybrid sensor placement approach ensures the quick detection of adverse events, thereby reducing their impact on the community. Another significant advantage of our approach is the flexibility provided by the adaptive monitoring capabilities of the combined in-situ and mobile sensor deployment.



(a) Average detection and localization times of the deployments for geo-correlated events

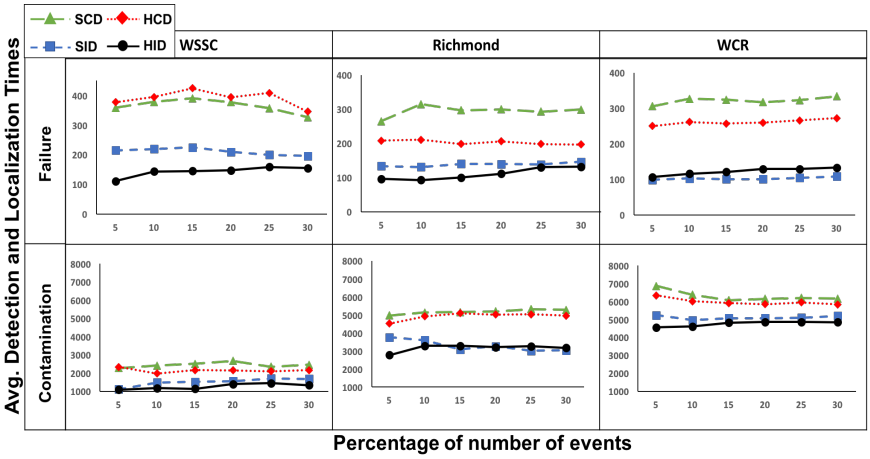


(b) Average normalized impact caused by geo-correlated events in the time taken for the deployments to detect and localize them

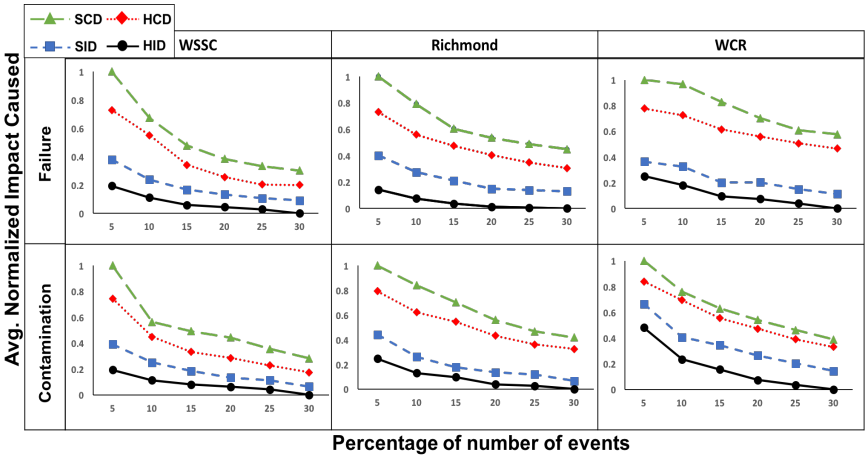


(c) Normalized Cost Effectiveness of the sensor deployments for geo-correlated events

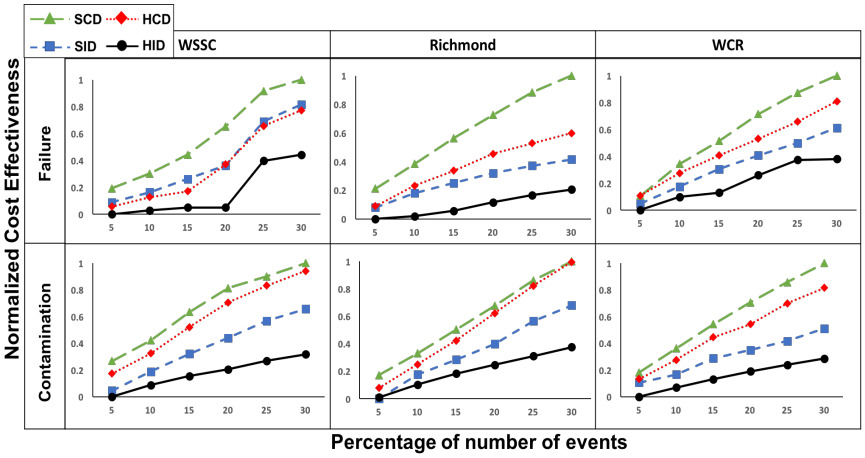
Figure 4.9: Comparison of Approaches for Geo-correlated events



(a) Average detection and localization times of the deployments for critical events



(b) Average normalized impact caused by critical events in the time taken for the deployments to detect and localize them



(c) Normalized Cost Effectiveness of the sensor deployments for critical events

Figure 4.10: Comparison of Approaches for Critical events

Chapter 5

Operational Framework for Resource Efficient Adaptive Monitoring

Once sensors are deployed, their data streams need to be transmitted to edge and cloud servers over communication networks for analysis using analytical models. However, these servers often have limited amount of resources using which they need to support multiple infrastructure monitoring applications. Moreover, the heterogeneity in devices, network links, and analytical models often results in multiple choices of monitoring pipelines or workflows that can be leveraged by a single application, each with their own benefits and incurred resource costs. Since community infrastructures are complex and in a state of continuous flux, developing a one-size-fits-all monitoring framework that works for all infrastructures and communities is infeasible. In this chapter, we discuss how this heterogeneity can be leveraged to provide adaptive monitoring in a resource-efficient manner.

Specifically, we develop an operational framework, named **REAM**, that determines the optimal choice of monitoring pipeline (sensors, network links, and analytical models) to execute at any given time, depending on the existing community structure and behavior. REAM

adapts to changing community and infrastructure conditions, and balances the monitoring requirements of each application (e.g.) accuracy, while judiciously utilizing the limited resources available at edge and cloud servers. We evaluate REAM on real-world infrastructure testbeds and our results indicate that the monitoring pipelines chosen by REAM's operational decision making maintain high monitoring performance while incurring significantly low resource consumption costs.

5.1 Chapter Overview

Infrastructure monitoring applications using IoT are ubiquitous across many domains like transportation, environmental sensing, power grid, water distribution networks, buildings, among others [142, 179]. These infrastructure can cover diverse geographical areas – classrooms, buildings, road intersections, city districts, etc. – and can be instrumented with varying density and heterogeneity of sensors. The monitoring applications often generate vast quantities of data, and while the collected data can be sent to cloud data centers or servers for analysis, this often leads to high operational costs, slow response times, and service interruptions, as these cloud servers are often far away from the community infrastructure being monitored [60]. Moreover, monitoring applications that are time-sensitive and critical (e.g., flood, fire detection), can suffer from significant performance degradation that can have a large negative impact on the community. An alternate solution is to leverage *edge servers* which can be network gateways or dedicated workstations, that are in closer proximity to the infrastructure [10] for less expensive, more responsive, and quicker analysis results. In addition, as we described earlier, many community agencies have privacy and security policies that prevent them from solely using a public cloud provider, and instead require solutions that can be deployed on servers located on-premise. Therefore designing an edge or hybrid cloud driven operational framework is important for monitoring applications run by these public works agencies.

Figure 5.1 illustrates a sample road intersection instrumented with various sensors. In this infrastructure, cameras, motion sensors, moisture sensors and traffic lights are connected to the edge server and power source via wired connections. Other sensors such as turbidity and pH are battery powered and wirelessly connected. Remote services like weather forecasts and social media reports can also be used at the edge to provide external information about the community. System administrators or community stakeholders may choose to concurrently execute different monitoring applications and hence activate different sensors and analytics

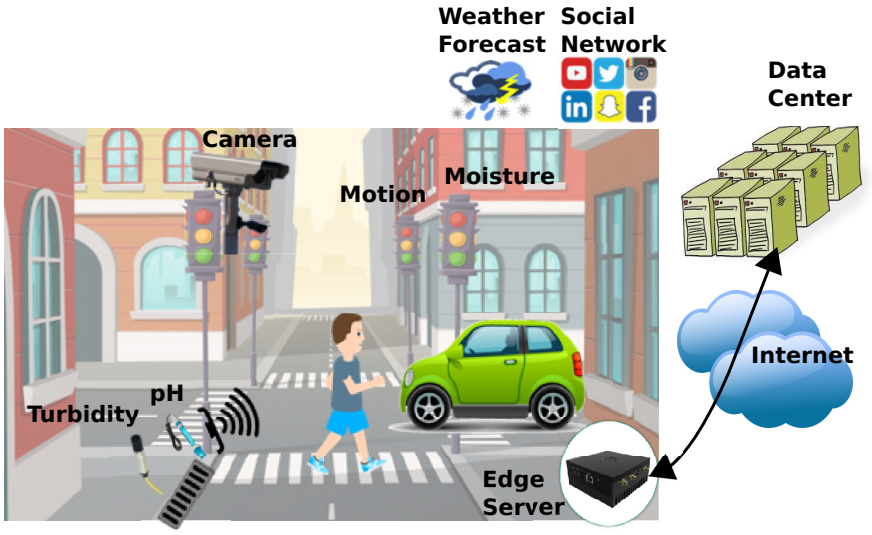


Figure 5.1: Sample use case of a community infrastructure instrumented with multiple sensors and actuators, an edge server, and other environmental contextual information.

at any given moment. Moreover, the quality of network links over which the sensor data and analytics results are sent can impact the effectiveness of the monitoring applications due to factors such as loss of data packets, low bandwidth availability, etc. Given that the applications, sensors, analytics, networks, and edge servers are all highly heterogeneous as described in Section 3.2, effective choices of sensors, network links, and analytical models in conjunction with efficient resource allocation is key to the success of an infrastructure monitoring framework.

The objectives of monitoring applications could be met using approaches that use different sets of sensors and analytic operators, each of which would require a certain amount of compute, networking and other resources, and would provide a certain quality of results for the application [38, 41]. For example, a pedestrian detection application could: (1) use video feeds from a surveillance camera and an object detection algorithm, or (2) set up a motion sensor to activate above a certain threshold. The first approach is *fine-grained* and provides more accurate results while incurring much larger costs in terms of compute and networking resources for continuous monitoring than the second approach which is more *coarse-grained*. Since the events driving most monitoring applications are not continuous

and can occur sporadically, the costs of utilizing resource-heavy sensors and analytics for continuous monitoring can quickly add up.

We propose a **Resource Efficient Adaptive Monitoring (REAM)** framework [162, 160] to dynamically select the sensors, network links, and analytical models to execute at any given time depending on the current state of the infrastructure. The framework also takes into account application priorities while ensuring low compute, networking and energy costs. Realizing REAM is no easy task due to the following challenges:

- **Interoperability:** Most community spaces are progressively deployed with high heterogeneity in the instrumented devices (and thereby data types), reliable communication networks, and available analytical models. REAM needs a communication model to ensure efficient and reliable data and information exchange between these different types of IoT devices and edge/cloud servers.

- **Flexibility:** The objective of each monitoring application can be met with different pipelines or workflows of sensors, network links, and analytics, each of which would require a certain amount of compute, networking, and other resources, and provide a certain quality of results [38, 41]. REAM therefore should leverage the different options of sensors, networks, and analytics, while meeting the quality requirements of each monitoring application.

- **Adaptability:** Community spaces are complex and dynamic, while events in different infrastructure can take place under different contexts due to differences in location, demographics, structure, etc. It is, therefore, extremely challenging to develop accurate rules or models for each individual space. It is also important for the framework to be able to make online decisions with noisy inputs and to work well under diverse network conditions and resource availability. Making adaptation decisions, therefore, is a key objective of our REAM framework.

To achieve *interoperability*, we adopt the publish-subscribe data exchange model to facilitate efficient information exchange. This provides a robust methodology for control and data flows within the REAM architecture, where sensors, and edge/cloud servers can communicate with one another allowing for community stakeholders to easily deploy and manage REAM in a distributed environment. We also partnered with Real Time Innovations [64] and use their software to implement publish-subscribe based data exchange in our REAM deployments.

Secondly, in addition to sensing and analytics, we also incorporate network quality awareness in REAM's decision making for better *flexibility*. Most real-world deployments have heterogeneous modes of connectivity such as WiFi, Bluetooth, etc., which offer varying levels of Quality-of-Service (QoS). Moreover, the quality of connectivity is not constant and can vary based on the network utilization and data transmission of other devices and applications. Incorporating network quality awareness is an important step towards a practical and more realistic approach to real-world deployments which we demonstrate through experimental evaluations. Most prior efforts to monitor community infrastructure with IoT assume each application only comes with a predetermined workflow of sensors, network links, and analytics [179, 138]. However, it is important to note that under certain contexts, coarse-grained approaches can provide sufficient quality results and can also be used to trigger fine-grained approaches. For instance, in the pedestrian detection example described above, the number of instances of pedestrians crossing an intersection on a quiet street during night time would be low. Hence, the coarse-grained motion sensor based approach could be used to detect the potential presence of pedestrians and to then trigger the fine-grained camera based approach if a pedestrian was detected. This *adaptive* approach would be able to achieve sufficient quality of results while incurring lower costs than if the fine-grained approach was run continuously. Infrastructure monitoring applications could attain sufficiently accurate results while incurring low costs by intelligently deciding between using different monitoring workflows at different times based on the state of the community space and other contextual information. This decision making framework can be implemented in different ways,

including simple heuristics, rule-based approaches and learning driven models. We ensure flexibility by enabling stakeholders to define their own workflows or monitoring pipelines, which also addresses situations where any component has to be added or removed.

For high *adaptability*, we leverage Reinforcement Learning (RL) [144] to develop a *decision making algorithm* for the most appropriate adaptation decisions at any given moment. Our RL-based algorithm employs *agents* to directly learn from experience by interacting with the diverse and dynamic community environments. We demonstrate how our design choices enable seamless integration of the RL agents with the publish-subscribe data exchange, as well as how the incorporation of network quality awareness is critical for improved decision making in real-world community spaces. We extensively evaluate REAM on real world testbeds and demonstrate its effectiveness across diverse applications and deployments, in addition to evaluating its scalability for large-scale deployments.

Specifically, we make the following contributions:

- Design a novel REAM framework that jointly considers IoT sensors, analytics operators, and network links at the edge when monitoring community spaces. (Section 5.2)
- Formulate a decision making problem for the REAM framework to make adaptation decisions under the constraints of resource availability and network quality. We then present our RL-based algorithm and show case two real monitoring applications: stormwater contamination monitoring and pedestrian counting, while many other monitoring applications are possible. (Section 5.3)
- Evaluate our REAM framework on two real-world testbeds in Orange County, USA and NTHU, Taiwan and compare it to baseline approaches, when assuming the network loss is negligible. We show that REAM can achieve $> 90\%$ monitoring accuracy while incurring $\sim 50\%$ lower resource consumption costs compared to existing static

monitoring approaches. (Section 5.4)

- Conduct detailed simulations to validate the network-quality awareness of our framework under different network loss rates as well as its scalability to larger-scale deployments. We demonstrate that REAM can achieve up to 42% improvement in accuracy over static approaches by being cognizant of and adapting to differing network quality conditions. We also show that REAM can provide near real-time service for a multitude of monitoring applications, and incurs only a minimal increase in runtime (< 0.002 seconds), for even large scale deployments with over 1000 nodes. (Section 5.4)

5.2 Adaptive Monitoring Framework

In this section, we describe the design choices and architecture of our proposed REAM framework. In order to optimize the selection of sensors, analytics operators, and their associated network links, we organize them into workflows. The choice of using RL agents over a supervised learning approach is due to the fact that community behavior and patterns often change over time and hence the definition of events in a space can change. It is also not always obvious what the right sensing and analytics option is and may require a sequence of selections, and hence RL provides more flexibility in the way an agent can be trained to adapt to changing conditions. In order to support the publish-subscribe data exchange described earlier, we develop a middleware at each edge server and also define a coordinator between edge servers.

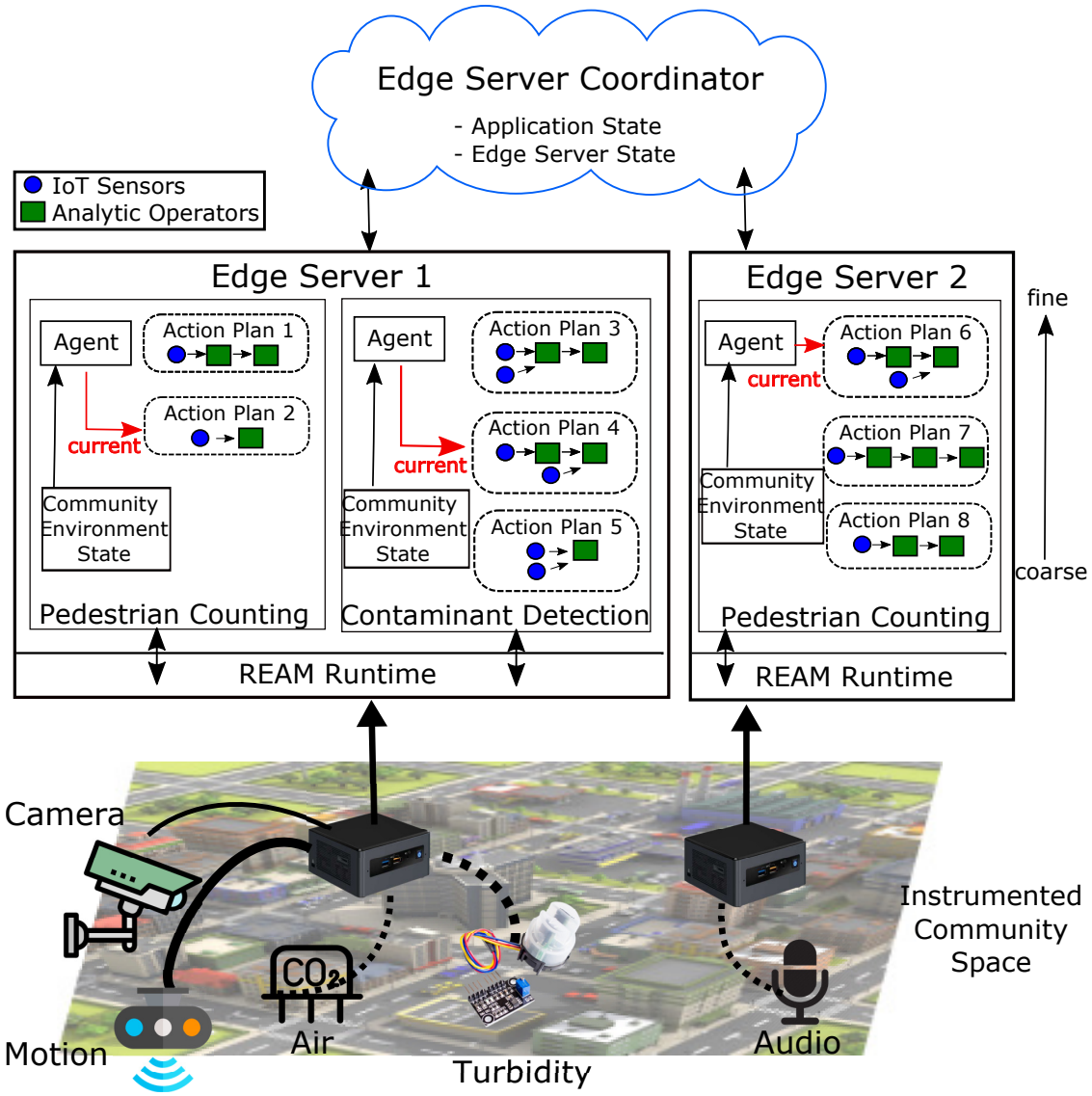


Figure 5.2: REAM framework architecture.

5.2.1 Architecture

Fig. 5.2 illustrates our proposed infrastructure monitoring framework with two edge servers that have three sample monitoring applications running on them. Each monitoring application relies on the measurements of a specific set of sensors that have been instrumented in the community infrastructure. The communication and data transmission between the sensors and the edge server can take place through various networks like WiFi, Bluetooth, ZigBee, LoRa, Ethernet, optical fiber, etc. These network links can have differing quality

levels captured by network loss rate, available bandwidth, etc, depicted by the dashed lines in Fig. 5.2. Once the sensor data are received at the edge server, they are run through a set of analytical models which could constitute ETL (Extract, Transforming, Load) functions, Machine Learning models, Time-series analyses, among others in order to obtain useful information. Next, we introduce several key software components in REAM architecture.

Action Plans. Since each application can use different combinations of sensors and analytics to achieve its objective with differing quality of results, we define each combination as an *action plan* where each plan can be thought of as a *workflow*, or an execution graph, of sensors, analytical models, and network links. They can vary in their execution complexity (large workflows with numerous sensor inputs and analytics), their resource requirements (resource-heavy sensors, large data volumes, complex analytics models), as well as in the quality of their network links (links with high loss rates would result in more packets being dropped). In our framework, as illustrated in Fig. 5.2, every monitoring application is a collection of action plans which can be a *coarse-grained* action plan that provides a baseline quality of continuous monitoring while consuming less resources, or various *fine-grained* action plans that provide a range of in-depth monitoring at higher costs, providing better results.

RL Agents. In the REAM framework, at each timestep, an application can choose to execute one of its action plans as denoted by the *current* tags in Fig. 5.2. The decision of which action plan to execute is taken by an RL agent that learns by interacting with the community infrastructure based on its application's objective. The agent observes the readings from the application's sensors, network link quality information, outputs of the analytical models, and other external community environment contextual information such as the weather and time-of-day. It uses this information to develop a probabilistic learning model that drives the selection of which action plan to execute based on the effectiveness or utility of each plan.

Our framework design assigns one agent for each application. We opt not to create a global agent across all applications at the edge for the following reasons:

- *Flexibility:* Individual agents simplify the process of dynamically adding or removing monitoring applications since agents can be trained independently of others unlike with a global agent.
- *Tractability:* The dynamic and complex nature of community infrastructure can result in the agent having to reason about an extremely large number of states [45]. By assigning one agent to each application, we can ensure that the number of states is manageable.
- *Simplicity:* Applications can have different objectives and operate at different time granularities. It is therefore difficult to define a global objective for each community infrastructure that is normalized across different applications. Furthermore, individual agents allow each application to set its own timestep granularity for sensing, monitoring, and analysis, even though the resulting decisions may slightly differ from the optimal ones.

REAM Runtime. In order to facilitate the exchange of information - both data and control messages, we implement a middleware on each edge server called REAM Runtime. As shown in Figure 5.3, the REAM Runtime hosts the publish-subscribe data exchange implementation of REAM, and handles the information flow of data from the community environment to the edge server, capturing the network and environment states. In addition, the REAM Runtime middleware also tracks the amount of resources available on the edge server which is an essential information for the RL agent to be able to decide on feasible action plans that it can utilize. Finally, REAM Runtime also communicates back to the devices and analytical models that have been selected by the RL agent using the publish-subscribe paradigm. We further detail the control and data flows in REAM in the next section.

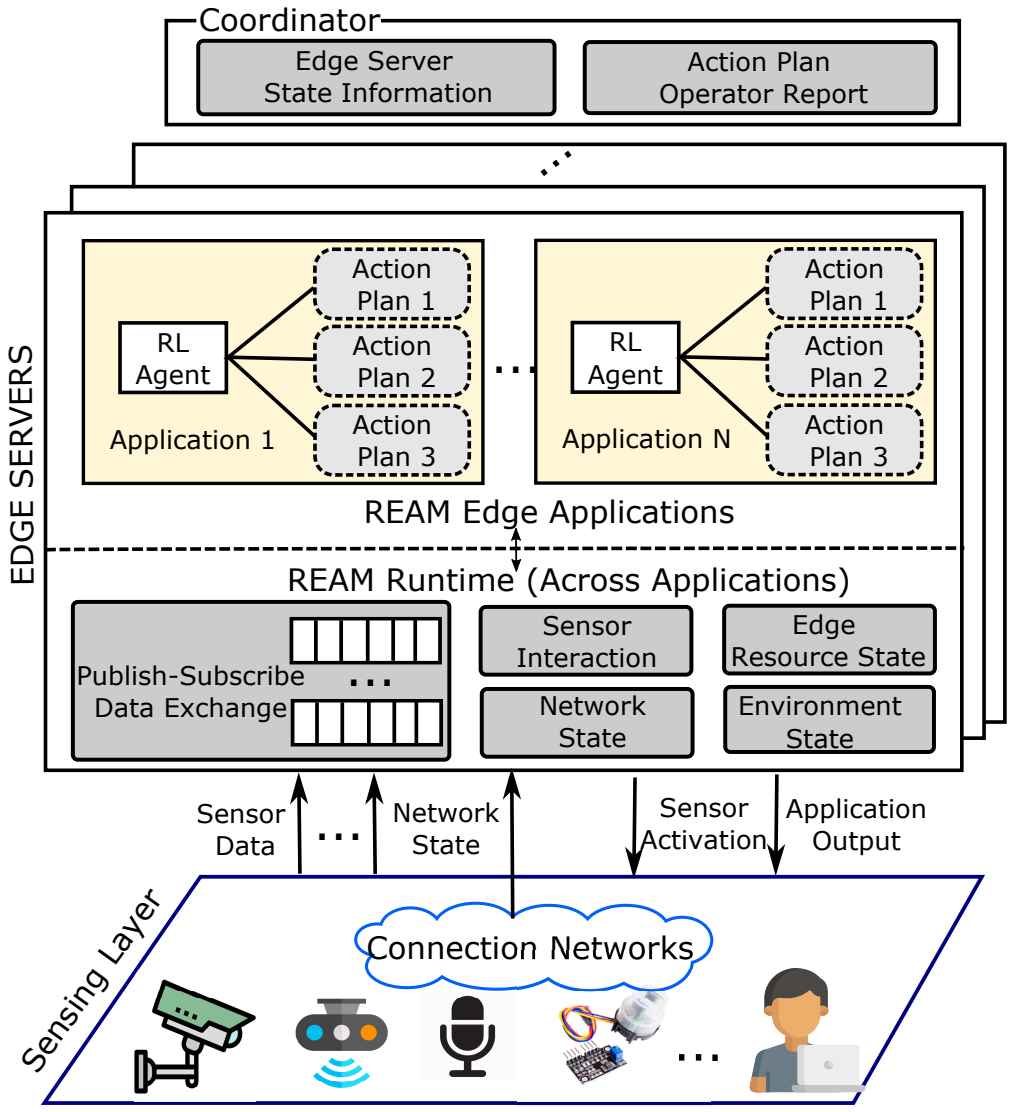


Figure 5.3: REAM uses a publish-subscribe data exchange model for control and data flows.

Edge Server Coordinator. In order to maintain a repository of the available action plans and the state of each edge server and its applications, we design an Edge Server Coordinator that can reside in the cloud or on an edge server. It also maintains knowledge of (1) the application states including their action plans, resource requirements, and objectives, and (2) the edge server states which include their resource availability, current applications, action plans, and corresponding network link quality. Agents can use such information to take decisions, and system administrators can use the coordinator to modify application objectives and resource availability.

5.2.2 Control and Data Flows

Fig. 5.3 presents the control and data flows in our REAM framework. In order to facilitate information and data exchange among the components, the REAM framework architecture uses a *publish-subscribe* data exchange model. This model allows sensors, edge servers, network links, analytical models, and the coordinator to have their own topics (or channels) that they publish information to, which can be accessed by subscribing to these topics. In particular, sensors publish their raw data streams and network link quality information to dedicated topics, which are subscribed to by edge servers hosting one or more analytical models that analyze the sensor's data. The computed analytics results by these models are then published to dedicated topics for each application, that can be subscribed to by community stakeholders and administrators. The action plans selected by the RL agents are also published in order to activate the appropriate sensors and models in the selected plan by the REAM Runtime. It also communicates with the coordinator to exchange resource availability, action plan selection, and metadata information.

5.3 Problem Formulation

In this section, we formulate the problem of resource efficient adaptive monitoring in community infrastructure, describe our approach to represent the decision making as an RL task, and then present our solution approach. We provide a list of symbols used throughout this article in Table 5.1.

Table 5.1: Symbols used in this chapter

Symbol	Description
$\mathcal{A}$	Set of monitoring applications
$\mathcal{S}$	Set of sensors
a_i^{ϕ}	Priority of application a_i
$\mathcal{O}$	Set of analytics
$\mathcal{L}$	Set of network links
$\mathcal{P}$	Set of action plans
$\mathcal{B}(p_j)$	Benefit of action plan p_j
$\mathcal{C}(p_j)$	Cost of action plan p_j
R_k	Amount of resource of type k
$\mathcal{N}(p_j)$	Network loss on links of action plan p_j
$\mathcal{U}(p_j)$	Utility of an action plan p_j
r_t	Reward at time step t
$\mathcal{S}'$	Operating state of the sensor servicing the application
$\mathcal{O}'$	Analytical models currently running
$v(\mathcal{A}')$	Value returned by the application's analytics
$\mathcal{N}'$	Loss rates of the network links used by the application
Ext	External contextual information about the community space
$\mathcal{P}'$	Set of valid action plans
J_t	Cumulative reward

5.3.1 Utility Driven Action Plan Selection

We consider a community space that has a set of monitoring applications $\mathcal{A}$, where each application $a_i \in \mathcal{A}$ has a priority a_i^{ϕ} associated with it. For example, a gunshot detection application in a community would have a higher priority than a parking violation monitoring application. The community space is instrumented with a set of sensors $\mathcal{S}$, whose data can be analyzed using a set of analytical models $\mathcal{O}$ that are transmitted over a set of network links $\mathcal{L}$ and are hosted on a set of edge servers.

We define a set of *action plans* $\mathcal{P}$, where each plan $p_j \in \mathcal{P}$ consists of a workflow of sensors, network links, and analytical models, and services a specific application. Each action plan p_j provides a certain *benefit*, $\mathcal{B}(p_j)$, for the monitoring application it services which is dependent on the application's objective. Each plan p_j also incurs a *cost* $\mathcal{C}(p_j)$ which reflects the amount of resources R_k of type k (e.g., CPU, bandwidth, power, memory, etc.), that it consumes to run all the sensing and analytics present in the action plan. Each plan also incurs a *network*

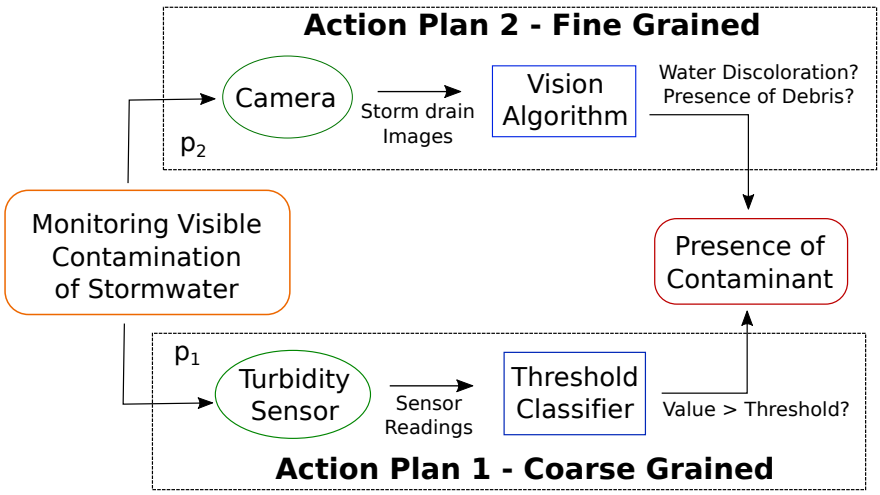


Figure 5.4: Example of two action plans for a stormwater visible contamination monitoring application.

loss $\mathcal{N}(p_j)$ reflecting the network loss rates across the links $l_i \in \mathcal{L}$, connecting the devices in the plan. We then define the overall utility of an *action plan* p_j as:

$$\mathcal{U}(p_j) = \frac{\mathcal{B}(p_j)}{\mathcal{C}(p_j)} \times \frac{1}{\mathcal{N}(p_j)}. \tag{5.1}$$

where the first term represents the tradeoff between the benefit and cost of each action plan, and the second term ensures that plans with high network loss rates reduce the overall utility of the plan.

Fig. 5.4 shows an example of two different action plans that service the same stormwater visible contamination monitoring application. Plan p_1 utilizes a simple turbidity sensor that would be less accurate than the camera based solution of plan p_2, since it relies on a manually set and potentially erroneous threshold. Moreover, today's state-of-the-art vision algorithms can typically achieve high levels of accuracy and thus p_2 can provide a much higher benefit to the application. However, the cost incurred by p_1 is much lower than that of p_2, since the periodic capture and transmission of images would consume a lot of network bandwidth, the camera would require more power, and the vision algorithm would also consume more

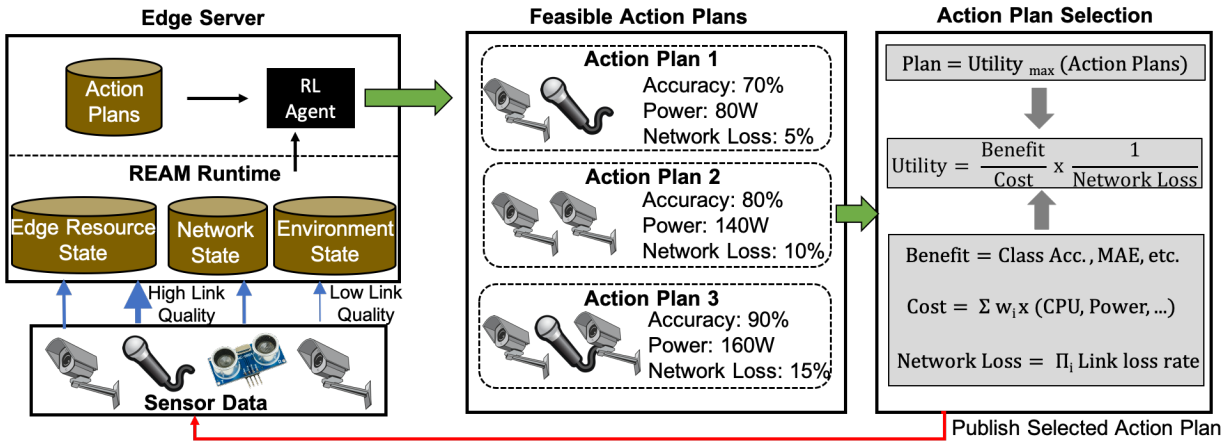


Figure 5.5: REAM action plan selection workflow.

compute resources in order to provide results in near real-time.

Moreover, the *benefit vs. cost* tradeoff captured by the utility of an action plan, is also dependent on various environmental contexts of the community space which REAM leverages. For instance, at night, the camera images may not be good enough for the vision algorithm to detect discoloration and debris, which might result in both action plans having similar accuracy. Hence, it would be a prudent decision to execute the coarse-grained plan more frequently at night since it can achieve similar benefit at lower costs, and the fine-grained plan during the day when it can provide much higher benefit. Furthermore, it is also important to consider the quality of the network links across which data is transmitted by each plan. If the links in plan p_2 result in a large number of packets being dropped, this can cause a drop in monitoring accuracy due to the lack of data delivery, even though the camera images provide high quality information, and hence using the turbidity sensor based plan p_1 might provide higher utility due to consistent data delivery.

We visualize the action plan selection workflow used by REAM in Fig. 5.5. Each edge server receives sensor data and network link quality information through the publish-subscribe data exchange model. The RL agent in the edge server then leverages this information, along with knowledge of the community environment states, current resource availability of the edge

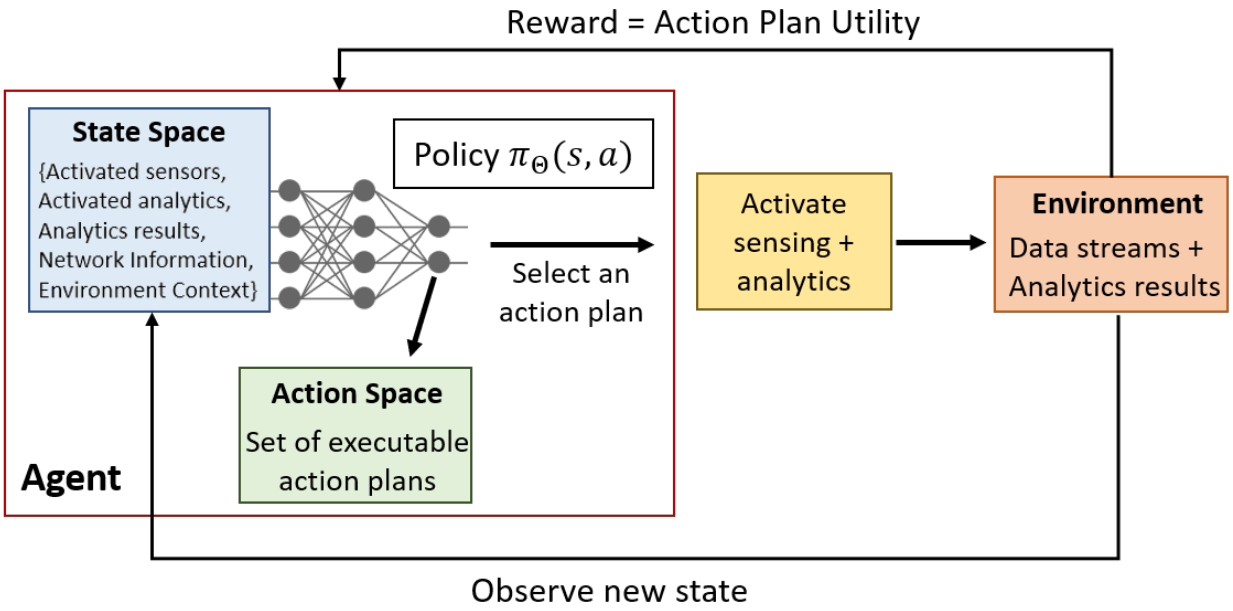


Figure 5.6: REAM RL agents selecting action plans following the policy learned by a Deep Neural Network (DNN)

server, as well as available action plans, in order to generate a set of action plan candidates along with their expected benefit and cost attributes. It then computes the relative utility of each plan and selects the plan with the highest utility. Once the plan has been selected by the agent, the REAM Runtime middleware then publishes this information to community stakeholders as well as to activate the sensors and analytical models in the selected action plan.

5.3.2 Defining RL Agents

Consider the general setting shown in Fig. 5.6, where an RL *agent* interacts with an *environment*. At each time step t, the agent observes some state s_t, and then chooses to perform an action a_t based on a policy. Once the action is performed, the environment transitions its state to s_{t+1} and the agent receives a reward r_t. The state transitions and rewards are stochastic and are assumed to have the Markov property, i.e., the state transition probabili-

ties and rewards depend only on the state of the environment s_t and the action a_t taken by the agent.

State Space. In the REAM framework, we represent the state s_t of each application's RL agent at any given time as a class object that consists of the following attributes - (1) $\mathcal{S}'$: the operating state of the sensors servicing the application, (2) $\mathcal{O}'$: the analytical models currently running, (3) $v(\mathcal{A}')$: the value returned by the application's analytics, (4) $\mathcal{N}'$: the network loss rates of the network links used by the application, and (5) Ext : external state and contextual information about the community space (e.g., time-of-day, weather information, etc.), which can influence the performance of the sensors and analytical models.

Action Space. At each timestep, an application's agent determines its action space as a set of valid action plans $\mathcal{P}' \subseteq \mathcal{P}$ that it could potentially execute from its current state. The timestep is configurable, which can be different for individual applications in the space. Each plan $p_j \in \mathcal{P}'$ consists of a set of active sensors, their operational states (on/off for simple sensors, pan–tilt–zoom for a camera), the network links used by devices in the plan, and a set of active analytical models together with its monitoring workflow or pipeline.

Reward. The reward r_t obtained by the agent for executing an action plan is the utility provided by that plan. The benefit of plan p_j depends on the specific application (e.g., classification accuracy, distance based error, etc.). We compute the cost of p_j by first normalizing the amount of resources required of each resource type (CPU, bandwidth, memory, etc.) across all action plans of the application and then calculating a weighted sum of these normalized costs for plan p_j as:

$$\mathcal{C}(p_j) = \sum_{k=1}^{|R|} w_k \times R_k^{p_j}, \tag{5.2}$$

where w_k refers to the weight and $R_k^{p_j}$ refers to the normalized amount of resources of type k

required for plan p_j. The weights allow system administrators to prioritize the conservation of certain types of resources and lessen their importance if they are abundantly available. We then compute the network loss for p_j as the multiplication of the loss rates across all the links, $l_i \in \mathcal{L}$ used by the plan:

$$\mathcal{N}(p_j) = \prod_{i=1}^{n} l_i \ , \ \forall l_i \in \mathcal{L} \tag{5.3}$$

5.3.3 Training RL Agents

Each agent can only control its action plan selection and has no apriori knowledge of the rewards or the state transitions which can be affected by external factors. During training, the agent interacts with the community space environment and observes the rewards and state transitions while choosing different action plans. The agent's goal is to select action plans in a way that maximizes the cumulative reward J_t it receives over any time period T:

$$J_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{5.4}$$

where γ is a discount factor $\in [0, 1]$ and $r_{t'}$ is the reward at timestep t'. We then define $Q^*(s, p_j)$ as the maximum expected reward achievable by following a policy $\pi(s, p_j)$, which refers to the probability of action plan p_j being chosen by the agent when in state s. That is:

$$Q^*(s, p_j) = \max_{\pi} \mathbb{E}[J_t | s_t = s, p_{j_t} = p_j, \pi] \tag{5.5}$$

. Using the Bellman equation [137], this can be represented as:

$$Q^*(s, p_j) = \mathbb{E}[r_t + \gamma \max_{p_{j_{t+1}}} Q^*(s_{t+1}, p_{j_{t+1}}) | s_t, p_{j_t}] \tag{5.6}$$

Algorithm 3 Deep Q-learning Algorithm

Initialize Replay Buffer $\mathcal{D}$
Initialize Q, DNN with random weights θ
for $t = 1 \rightarrow T$ **do**
    With probability ϵ, select a random *action plan* p_j otherwise, select $p_j = \max_p Q(s_t, p; \theta)$
    Communicate chosen plan with REAM Runtime and receive allowed plan p'_j
    Execute *action plan* p'_j and observe environment to get reward r_t and state s_{t+1}
    Store transition $(s_t, p'_j, r_t, s_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions (s_k, p_k, r_k, s_{k+1}) from $\mathcal{D}$
    Set $y_j = r_j + \gamma \max_p Q(s_{t+1}, p; \theta)$
    Perform gradient descent step on $(y_j - Q(s_t, p_j; \theta))^2$ with respect to θ

Since community spaces are complex and can have a large number of possible {state, action plan} pairs, it would be infeasible to store the policy $\pi(s, p_j)$ in a tabular form as a lookup table, also known as a Q-table. Instead, it is more common to use *function approximators* to represent the policy by estimating $Q^*(s, p_j)$. Among the approximators, Deep Neural Networks (DNNs) [54] have recently gained popularity for solving large-scale RL tasks since they do not need hand-crafted weights. A network with weights θ can be trained by minimising a sequence of loss functions $L_i(\theta_i)$ at each iteration i, where:

$$L_i(\theta_i) = \mathbb{E}\left[\left(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) - Q(s_t, a_t; \theta_i)\right)^2\right] \tag{5.7}$$

We represent the action plan decision making policy as a neural network with weights θ which takes the current state of the RL agent as input and outputs a probability distribution over all valid potential next action plans. Note that we allow the RL agent to continue executing the current action plan in the next timestep as well.

We train the agents using the deep Q-learning algorithm [99] as shown in Algorithm 3. It uses an ϵ-greedy policy [144] in order to select an action plan by either randomly selecting a

plan p_j with a probability ϵ, or selecting the plan with the maximum value of the probability distribution. At each timestep, the chosen action plan is executed and its reward and the next state are observed. We store the agent's transitions in a buffer $\mathcal{D}$ of a fixed size and then perform gradient descent to update the weights θ of the neural network (Line 10 in Algorithm 3) using a minibatch of transitions drawn at random from the buffer as:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}\Big[\big(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{i-1}) - Q(s_t, a_t; \theta)\big)\nabla_{\theta_i} Q(s_t, a_t; \theta_i)\Big] \tag{5.8}$$

Rather than computing the full expectations in the above gradient, it is often computationally expedient to optimise the loss function using approaches like stochastic gradient descent (SGD), Adam [73], RMSProp [147], etc.

5.3.4 Prioritized Action Plan Selection

Before the RL agents identify the optimal action plans, the edge server must perform a sanity check to find the subset of all action plans that can be feasibly executed without saturating all the available resources. That is, the REAM Runtime middleware at each edge server employs our proposed Prioritized Action Plan Section (PAPS) algorithm, as shown in Algorithm 4, to coordinate with the RL Agents in selecting action plans for varying resource availabilities and publishing that information to activate the appropriate sensors and analytical models in the selected plans.

More specifically, at each timestep, the REAM Runtime sequentially communicates with each application's RL agent in the order of their priority $\mathcal{A}^{\phi}$. It limits the candidate action plans for each agent ($\mathcal{P}' \in \mathcal{P}$) based on the current resource availability. The agent then selects an action plan with the highest utility (Equation 5.1), based on the current state

Algorithm 4 Prioritized Action Plan Selection

1: **Input:** Action Plans $\mathcal{P} = \{p_1, ..., p_n\}$, Plan resource requirements $\mathcal{P}^{\mathcal{R}} = \{p_1^R, ..., p_j^R\}$, Application priority $\mathcal{A}^{\phi} = \{a_1^{\phi}, ..., a_j^{\phi}\}$, Available edge server resources $\mathcal{R} = \{\mathcal{R}_1, ..., \mathcal{R}_k\}$
2: **for** $t = 1 \rightarrow T$ **do**
3: **for** $\phi = 1 \rightarrow j$ **do**
4: Pass action plan candidates $\{\mathcal{P}' \in \mathcal{P} \mid (\mathcal{P}')^R > \mathcal{R}\}$ to agent a_i^{ϕ}
5: Obtain action plan $p_j \in \mathcal{P}'$ with max. utility (Eq. 5.1)
6: $\mathcal{R} = \mathcal{R} \setminus p_j^R$
7: Publish to activate sensors and operators $\in p_j$

of the community environment. The REAM Runtime then obtains the selected action plan p_j, and updates the available resources ($\mathcal{R}$) as well as publishes this information to activate the sensors and operators that are parts of the selected plan p_j. If an edge server has limited resource availability, the algorithm ensures that high priority applications can select action plans with high utility, while potentially limiting the plans selected by low priority applications.

At every timestep, Algorithm 4 must handle all monitoring applications in the environment ($|A|$), each of which can have $|P|$ number of action plans. This can result in the need to publish to $|E|$ number of sensors and operators where E represents the total set of sensors and operators in the environment. This results in a polynomial time complexity of $O(PAE)$.

To illustrate our RL-based approach and the selection of action plans by the REAM RL agents, we give an example in Figure 5.5. We see that the edge server receives sensor data from heterogeneous devices across network links with varying quality. Algorithm 4 is used to determine the available resources on the edge server, and to determine the feasible action plans as shown in the third box. We can see that the three available action plans have varying benefits ($\mathcal{B}$), costs ($\mathcal{C}$) in terms of power consumption, and network loss ($\mathcal{N}$). To determine the reward of each plan, we normalize the values and use Equation 5.1 to obtain a utility of [28.0, 9.1, 5.9] for the three action plans respectively. The RL agent hence chooses the first plan and publishes it. The sensors and analytics in the selected plan are then activated and this process repeats for the newly observed state from the sensors in the selected plan.

5.4 Experiments

In this section we evaluate the resource-aware infrastructure monitoring capabilities of our proposed REAM framework and compare its performance to existing approaches. We present two monitoring applications in real-world testbeds located in Orange County, USA and National Tsing Hua University (NTHU), Taiwan.

In our first set of experiments (Sections 5.4.2 and 5.4.3) we evaluate the benefit vs. cost tradeoff in REAM. We assume that the network loss in these experiments are negligible. For each application, we compare the performance of REAM with static monitoring approaches that execute specific action plans in isolation. We also compare with a machine learning approach using random forest that uses the same training and test data.

We then evaluate how REAM performs in community environments with network loss. Since a controlled introduction of loss in the real community infrastructure testbeds is challenging, we simulate the input to the REAM prototype under conditions of network loss. We compare its performance with a network agnostic planning approach to evaluate the impact that network quality aware monitoring has on the monitoring accuracy (Section 5.4.4). We also evaluate the scalability of REAM to large-scale community deployments (Section 5.4.5).

5.4.1 Experimental Setup

Prototype Implementation. We implemented our REAM prototype shown in Fig. 5.3 using Python. The RL agents are implemented using Keras [35], where each agent is a neural network containing two fully connected hidden layers with 24 neurons. We update the policy network parameters using the Adam algorithm [73] with a learning rate of 10^{-3} and implement our analytic operators for monitoring applications using Scikit-learn [110].

For the data exchange implementation in the prototype, although there are many publish-subscribe protocols available (e.g., AMQP, MQTT), the REAM prototype implementation uses the *Data Distribution Service* (DDS) protocol which offers several advantages. Unlike other protocols, DDS does not require centralized brokers and instead allows for fully decentralized, data-centric, and peer-to-peer communications. DDS supports dynamic discovery of publishers, subscribers, and data transmission while offering 30+ QoS levels for tuning data exchange performance, resource usage, priority, reliability, etc., which allows REAM to be fine-tuned for various environments. DDS also supports UDP for both high-performance and multicast communications. There are various DDS libraries available like OpenDDS, Opensplice, Fast DDS, etc [85, 13, 119]. We chose to implement DDS in our prototype using the Real Time Innovations (RTI) Connext DDS framework [64] which provides specific implementations for resource-limited devices that are often found in smart community infrastructure deployments.

Our prototype implementation defines publish-subscribe topics for each sensor and analytics output, coordinator messages, and every edge server using the `DDSTopic` interface found in the RTI Connext API [65]. Each topic is listened to using the `DDSTopicListener` interface, and also published to by implementing an object of `DDSPublisher` class associated with each sensor and analytics operator. The network monitoring through DDS is done using the `DDSFlowController` interface that also allows packets to be sent using a fixed rate or even on demand. Each edge server has an instance of such a network monitoring component implemented in the REAM Runtime middleware.

Resource Measurements. Since the goal of the REAM framework is to be able to achieve application objectives while utilizing as little resources as possible, we capture the actual resource consumption (CPU, networking, and power) of the various devices and analytical models in order to run faithful experiments when comparing our solution against baseline approaches. Tables 5.2 and 5.3 summarize the resource consumption of individual devices

Device	Make/Model	Power (W)	Note
Motion	Optex LX-402	0.33	
Camera	LiteOn 3MP	3	
PC	Intel i3 @ 1.7 GHz	6	Idle
PC	Intel i3 @ 1.7 GHz	27.5	Loaded
Stormwater	In-Situ 600	0.54	

Table 5.2: Power Consumption of Sensors Deployed in the Infrastructure Testbeds

Analytics	CPU Usage	Memory Usage	Running Time per Frame
OpenCV	194.80%	1.7%	0.052s
YOLOv3	100.43%	8.5%	17.38s

Table 5.3: Resource Consumption of Video Analytics Models

and analytics in both testbeds.

5.4.2 Smart Stormwater Infrastructure

We utilize five stormwater *sensing units* that have been instrumented by Orange County Public Works Department (OCPWD) in order to monitor the quality of the water flowing through the storm drains, which is depicted in Figure 5.7. The stormwater can get contaminated while flowing into the drains by collecting pollutants like bacteria from human or animal waste, fertilizers, and even chemicals from industries that illicitly discharge their waste into these drains [154]. Each sensing unit consists of several hydraulic and chemical sensors to measure pH, turbidity, dissolved oxygen, flow rate, etc., that together are capable of detecting a wide range of potential contaminants. The sensor measurements are transmitted using LoRa networks and are analyzed at an edge server using Machine Learning classifiers to determine the presence of contamination. The sensing units are deployed in secure underwater housing and are battery powered. Accessing these units in order to replace the batteries, therefore, involves significant efforts to dig up the housing and access the hardware within, hence frequent battery replacement would incur large costs. OCPWD's

Figure 5.7: Our testbed in Orange County, USA: a storm drain and the locations of sensing units.

objective is to prolong the battery life while maintaining contamination event detection accuracy.

Since stormwater contamination events occur sporadically with long periods of normal activity, measurements of a subset of sensors can be sufficient to provide coarse-grained signatures that can then be used to trigger all the sensors for fine-grained monitoring during contamination events. This is because using all the sensors for continuous monitoring would consume a lot of battery power. The goal of deploying our REAM framework is to accurately identify stormwater contamination events while prolonging the battery life of these sensing units by appropriately switching between coarse and fine grained monitoring.

Stormwater Data

We use four months of sensor measurements from the sensing units at two different locations (A, B). For each location, we use three months of data for training and one month for testing. The measurements have a granularity of 15 minutes and the contamination event ground truth was annotated by an expert from OCPWD. We also obtained precipitation data for the location and battery consumption information of the sensing unit. We use two sensors - dissolved oxygen and pH, that are most sensitive to changes in the ecosystem to form the *coarse-grained* baseline action plan along with a Support Vector Machine (SVM) classifier. But since the changes can be due to minor natural variances in the chemical composition of the water, the coarse-grained plan can result in a number of false-positives. We hence define one *fine-grained* action plan that uses more information, consisting of the previous sensors along with temperature, Total Dissolved Solids (TDS), conductivity, and turbidity sensors and uses a Random Forest classifier, that is triggered by the coarse-grained approach and can more accurately determine if a contamination event has occurred. We define the reward for the REAM RL agent based on the utility provided, where the benefit $\mathcal{B}(p_j)$ of every action plan is its classification accuracy and its cost $\mathcal{C}(p_j)$ is the total battery consumption of the sensors and analytic operators in the action plan.

Stormwater Contamination Monitoring Results

We measured the accuracy achieved in classifying contamination events for the test data from both locations (Table 5.4 and Table 5.5). We observed that REAM achieved 90.9% and 80.1% accuracy at location A and B respectively, which is comparable to the 95.4% and 86.1% achieved by using only the fine-grained action plan and better than the 88.2% and 76.9% obtained by using the Random Forest supervised learning approach and the 73.3% and 68.6% achieved by using just the coarse-grained action plan. Moreover, the REAM

Comparison approach	Accuracy (%)	Total energy consumption (J)	Exp. battery life (days)	Avg. detection delay (mins)
REAM	90.9	86.40	53	20.2
Random Forest	88.2	101.77	44	24.7
Fine-grained	95.4	155.52	29	14.4
Coarse-grained	73.3	46.08	98	45.9

Table 5.4: Stormwater Contamination Monitoring - Location A

framework consumed on average 42% less energy than the fine-grained action plan across both locations, resulting in a longer battery life by 24 days at location A and 19 days at location B that we derived based on the two D-cell alkaline battery capacity of the In-Situ 600 stormwater sensing unit.

At location A, REAM had a 20 minute delay on average in detecting contamination events, compared to the 14, 24 and 45 minute delays achieved by the fine-grained, Random Forest, and coarse-grained approaches respectively. At location B, REAM's average detection delay was 14 minutes, compared to 10, 19 and 24 of the fine-grained, Random Forest and coarse-grained approaches respectively. Fig. 5.8 shows a zoomed in view of the contamination event ground truth and the action plans chosen by the REAM RL agent during a week of the test period. We observe that for most of the contamination events, the agent utilizes the fine-grained action plan to achieve high accuracy and ends up using the coarse-grained plan during periods when no events occur. The occasional shift to the fine grained plan as shown by the red circle occurs since the agent explores different action plans based on the ϵ-greedy policy described in Section 5.3.2 to adapt to changing environmental conditions, e.g., dry vs. wet weather, seasonal patterns, etc.

From these results, we can see that the REAM framework can increase the battery replacement cycle from less than 1 month with the fine-grained approach to almost 2 months with less than a 5% drop in accuracy and a detection delay within 5 minutes on average.

Comparison approach	Accuracy (%)	Total energy consumption (J)	Exp. battery life (days)	Avg. detection delay (mins)
REAM	80.1	95.76	46	14.5
Random Forest	76.9	120.93	35	19.3
Fine-grained	86.1	163.44	27	10.1
Coarse-grained	68.6	54.21	90	24.6

Table 5.5: Stormwater Contamination Monitoring - Location B

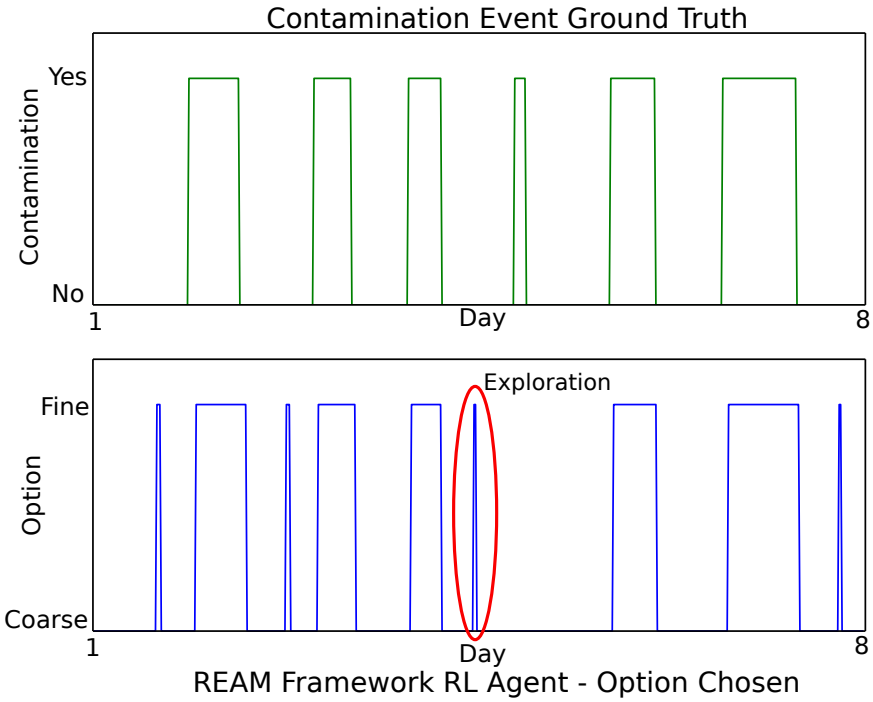


Figure 5.8: ϵ-greedy approach to selecting action plans by the REAM RL agent to determine stormwater contamination events

5.4.3 Smart Campus

We have instrumented eight smart street lamps on the NTHU campus in Taiwan, as shown in Fig. 5.9 for smart campus applications. In our testbed, each street lamp is instrumented with a power supply, an Ethernet switch, a Raspberry Pi (which also serves as a Bluetooth and Zigbee gateway), and a wide spectrum of environmental sensors, such as motion (PIR, passive infrared), temperature/humidity, and air quality (PM 2.5) sensors. Four of the

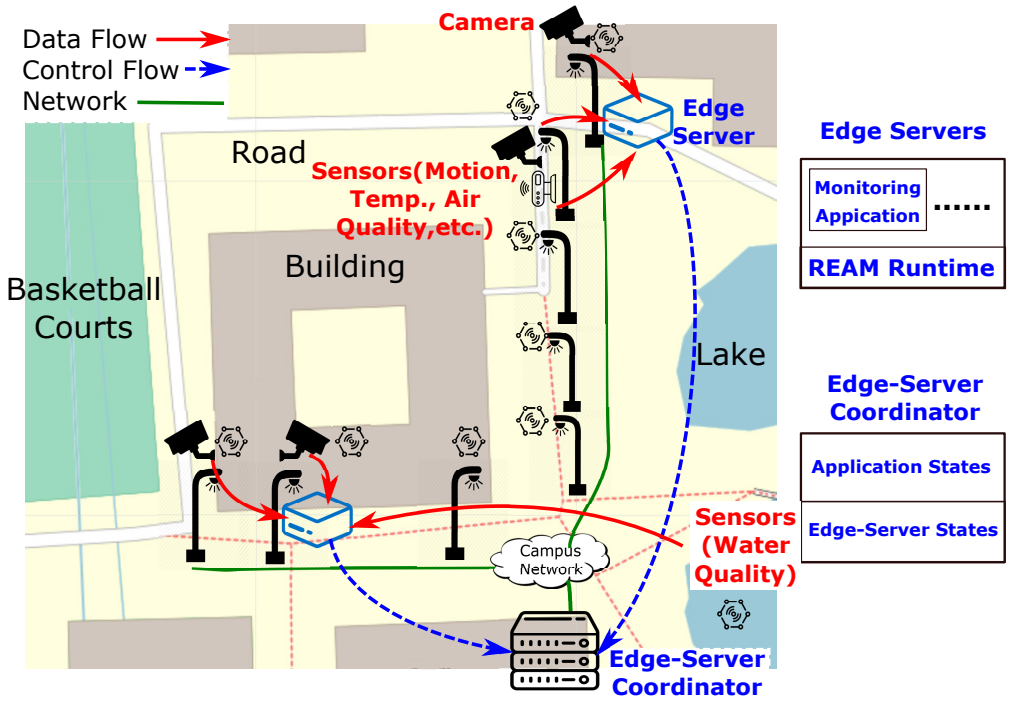


Figure 5.9: Our testbed at NTHU campus, Taiwan.

Figure 5.10: Sensor-rich smart street lamps (left) and a motion sensor (right).

lamps are equipped with 3MP cameras, of which three are fixed bullet cameras and one is a Pan–Tilt–Zoom (PTZ) camera. There's also an outdoor motion sensor installed on one of the

lamps, providing longer sensing range (12m) for the intersection. Fig. 5.10 gives the photos of our NTHU testbed. The lamps are connected using a heterogeneous network consisting of Gigabit Ethernet, WiFi mesh, LoRa, and NB-IoT. We install edge servers in two of the street lamps for running monitoring applications. The edge servers are Intel NUC PCs, each has a 4-core CPU at 1.7 GHz, 8 GB RAM, and 500 GB disk.

We utilize this testbed for a pedestrian counting application that attempts to profile the movement of people at main intersections. This is to dynamically dispatch security guards to direct on-campus vehicles when intersections are crowded. The goal of the campus administration is to infer these profiles using as little resources as possible to ensure availability for other on-demand (emergency) applications. Using fine-grained camera feeds coupled with analytic libraries like YOLOv3 [124] and OpenCV [22] can result in accurate counts, but this approach is resource intensive. Since the flow of pedestrians is not continuous (fewer people walking at night), a coarse-grained motion sensor could be used to trigger the camera based analytics in order to conserve resources. However, since different moving objects (e.g., car, bicycle, etc.) can also activate the motion sensor, its accuracy would be lower than that of camera feeds. The goal of deploying our REAM framework is to be able to learn when pedestrians are likely to be present and switch between coarse- and fine-grained monitoring to preserve resources.

Pedestrian Flow Data

For the pedestrian counting application, we use two different sets of data collected over different time periods. The first dataset (termed Data1) consisting of video data from a camera and a motion sensor on a street lamp overlooking an intersection, was obtained over a week in April. The second dataset (termed Data2), utilized three week's worth of video and motion sensor data from the same street lamp during August. Using two such datasets also allows us to evaluate the effectiveness of REAM during extended periods of differing

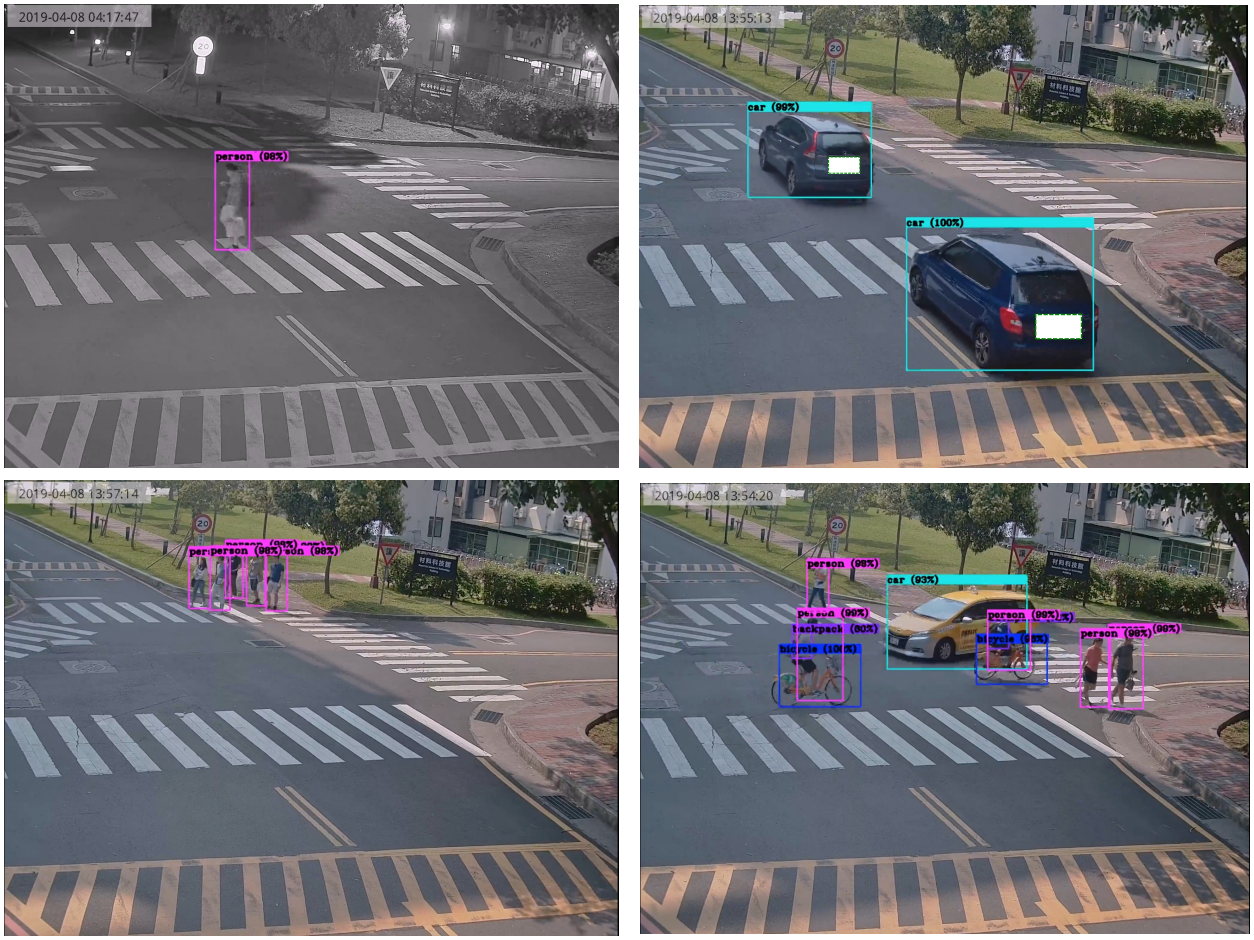


Figure 5.11: Sample video frames from a street lamp camera for the pedestrian counting application. The recognized objects and bounding boxes are given by YOLOv3.

pedestrian flow. For Data1, we use five days for training and two days for testing, while for Data2 we use ten days for training and the remainder for testing. The measurements have a granularity of one second.

Fig. 5.11 shows four sample frames of the video data. For both datasets, we define the *coarse-grained* action plan to consist of the binary output analyzed from the motion sensor. This plan is sufficient to capture situations where there are none or just one pedestrian at a given time as shown in the top left frame of Fig. 5.11. However, we notice that the motion sensor can be triggered by other objects such as the vehicles in the top right frame resulting in false positives. Hence, we define two *fine-grained* action plans that run OpenCV [22] and

Comparison approach	Distance from ground truth (%)	Total power consumption (W)	Total data generated (GB)
REAM	7.1	61.8	33.1
Random Forest	15.4	54.6	30.3
YOLOv3	0	126.5	55.62
OpenCV	37.3	39.32	55.62
Motion Sensor	62.3	36	0.0005

Table 5.6: Pedestrian Counting–Data1 (April)

Comparison approach	Distance from ground truth (%)	Total power consumption (W)	Total data generated (GB)
REAM	8.6	319.9	160.5
Random Forest	14.1	291.3	141.6
YOLOv3	0	697.5	305.9
OpenCV	32.1	246.2	305.9
Motion Sensor	54.9	198	0.006

Table 5.7: Pedestrian Counting–Data2 (August)

YOLOv3 [124] object detection algorithms respectively, which can also handle cases where there are many pedestrians simultaneously present as shown in the bottom left frame. The YOLOv3 library is more powerful in that it can more accurately handle situations where there are multiple different objects like pedestrians and vehicles present together as shown in the bottom right frame. We hence assume that the output of the YOLOv3 plan is the ground truth for our evaluations. The benefit of every action plan is defined as its distance from the ground truth in terms of the pedestrian count, and its cost is a weighted sum of its power, bandwidth, and CPU consumption. We assume equal weights in the evaluations if not otherwise specified.

Pedestrian Counting Results

Tables 5.6 and 5.7 show a summary of the performance comparisons, where we report the total power consumption as a sum of the power consumption of the sensors (motion, camera) and the edge servers. REAM achieved pedestrian count errors of 7.1% and 8.6% for Data1

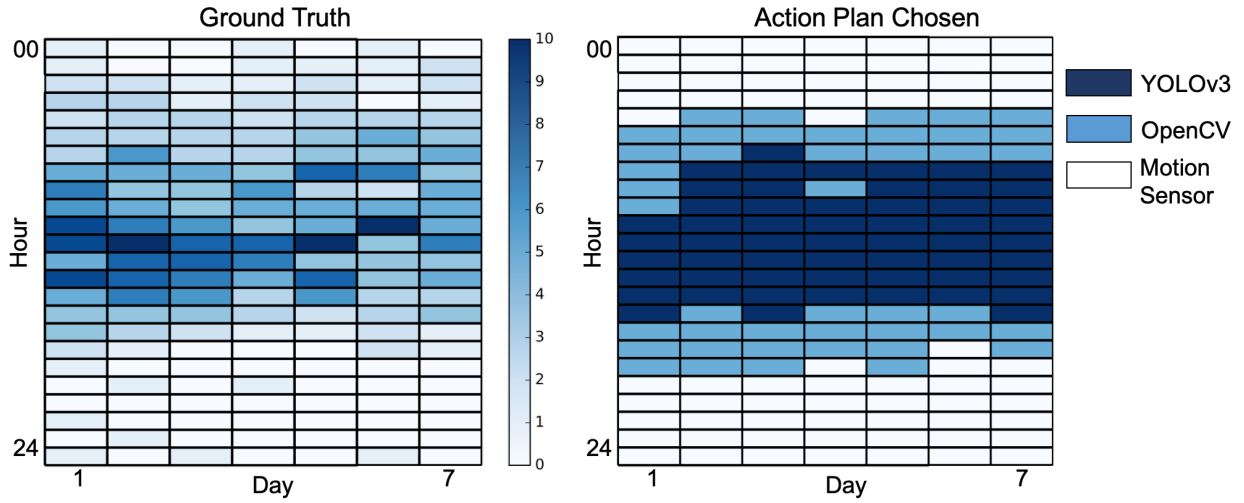


Figure 5.12: Comparisons of hourly pedestrian count ground truth and action plan chosen by REAM framework during a week – Data1 (April).

and Data2, respectively, compared to the YOLOv3 based approach that we assumed to be the ground truth and performed better than the Random Forest, OpenCV, and the coarse-grained motion sensor based approaches for both Data1 and Data2. However, the YOLOv3 library is very resource intensive, and this coupled with the significant power consumption of using a camera continuously, results in REAM having 51.0% and 54.1% less power consumption over the two datasets. The REAM framework also results in 44.0% and 47.5% less data being generated than the YOLOv3 and OpenCV approaches that require continuous generation and transmission of video data.

Figures. 5.12 and 5.13 illustrate heatmap based comparisons of the ground truth of average hourly pedestrian flow per week and the corresponding most frequent action plan chosen by the REAM RL agent during that hour for Data1 and Data2, respectively. While we observe that the flow of pedestrians was slightly lower in Data2 (August) compared to Data1 (April), in both cases, the number of pedestrians is the highest during the day (8am - 5pm) and during those periods the predominantly used action plan is YOLOv3 which results in high accuracy. Also, during the night and early mornings, when extremely few pedestrians are on the road, the RL agent chooses to use the motion sensor approach which is sufficiently accurate to

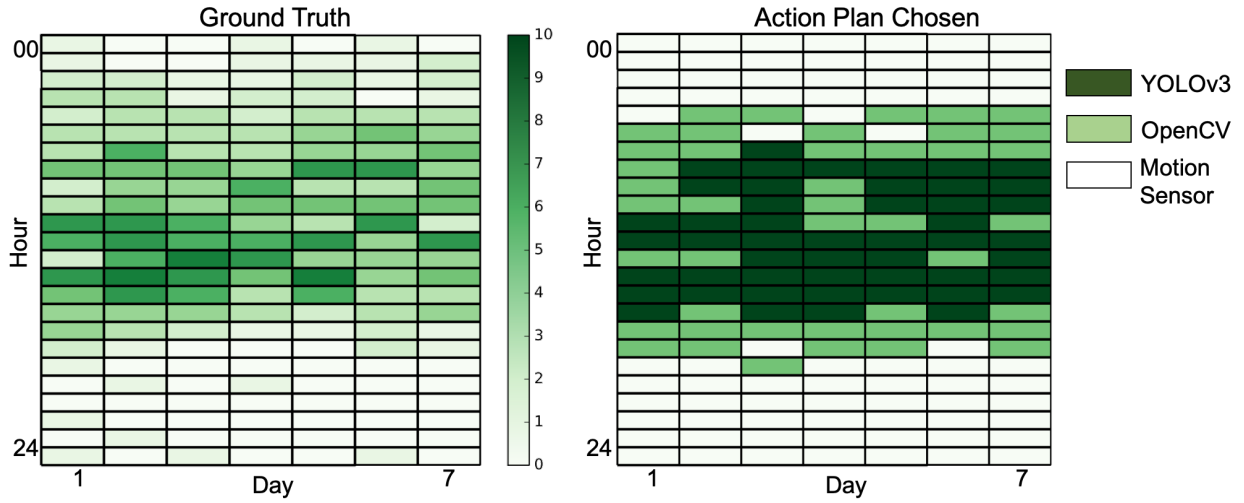


Figure 5.13: Comparisons of hourly pedestrian count ground truth and action plan chosen by REAM framework during a week – Data2 (August).

model the pedestrian flow. The REAM framework can thus achieve $> 90\%$ accuracy (hence not missing many people), while consuming $\sim 50\%$ less power and generating less data (hence consuming less resources), compared to static monitoring approaches. The performance of the RL agent could be made better by including additional contextual information like the weather, campus holidays, etc., that would have a direct impact on the pedestrian flow.

5.4.4 Exploring Network Loss Performance

We next evaluate the performance of our REAM prototype in the presence of network loss. Due to the challenge of experimenting with controlled network loss in a real community infrastructure, we simulate the input and network loss to the prototype based on our testbeds and compare the performance of REAM with a network agnostic planning approach. We use images from camera sensors with two different analytic operators YOLOv3 and OpenCV. As described earlier, while YOLOv3 achieves higher levels of accuracy, it uses more resources in terms of power consumption, CPU, and memory, as compared to OpenCV. In addition, we also leverage a feature in RTI's DDS framework to track the number of messages lost or

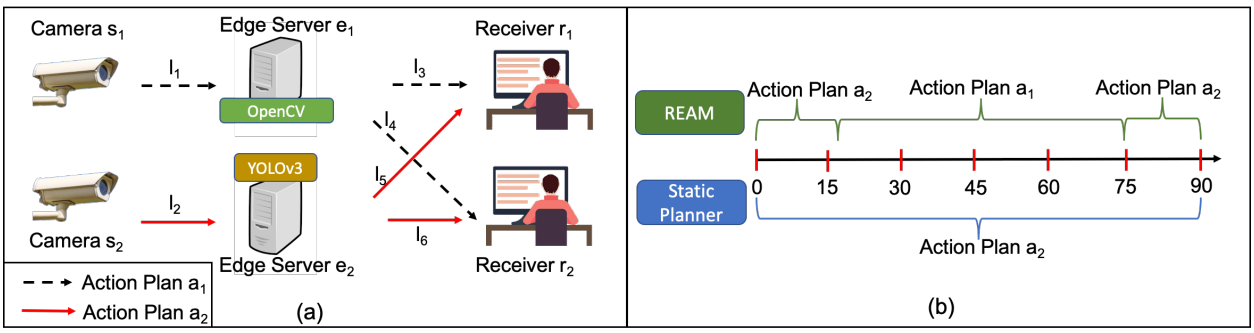


Figure 5.14: Network quality simulator: (a) experiment setup and (b) action plans selected by REAM and the static planner.

received across all action plans over time.

In order to evaluate REAM under conditions with network loss, we define two action plans as shown in Figure 5.14(a). The setup consists of 2 camera sensors $\{s_1, s_2\}$, 2 edge servers $\{e_1, e_2\}$, and 2 receivers $\{r_1, r_2\}$, with network links $\{l_1, ..., l_6\}$ connecting them as shown in the figure. There are 2 action plans $\{a_1, a_2\}$ that can be chosen, where a_1 uses OpenCV, and a_2 uses Yolov3 as the analytics operators respectively and hence benefit $\mathcal{B}(a_2) > \mathcal{B}(a_1)$. We assume that the sensors and edge servers are homogeneous which means that the cost $\mathcal{C}(a_2) = \mathcal{C}(a_1)$. When each experiment begins, the nodes start publishing data, and after 15 seconds, we introduce network loss on Link l_2. The loss rate persists for 60 seconds, after which it is set back to zero.

We compare the performance of REAM's network aware action plan selections with a static planning approach that uses Benefit and Cost to determine the action plan to use, but is agnostic to variability in network link quality, i.e., $\mathcal{U}_{\text{static}} = \mathcal{B}(p_j)/\mathcal{C}(p_j)$. Figure 5.14(b) shows the action plan selections during the experiments by both approaches. We see that for the initial period with no network loss, both approaches choose action plan a_2 which has higher utility than a_1, since its benefit is higher and the cost and network loss is the same. However, when the network loss is introduced on Link l_2, REAM uses this information and selects a_1, whose utility becomes larger due to the network loss. However, the static

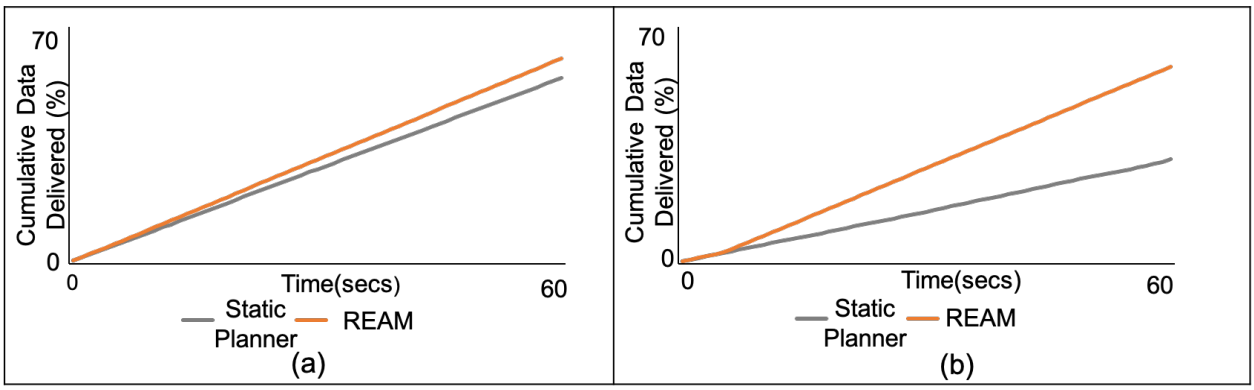


Figure 5.15: Cumulative data delivered by REAM and the static planner for network loss rates of: (a) 0.1 and (b) 0.5, respectively

planning approach continues to use a_2. When the network quality is restored, REAM once again switches back to action plan a_2 whose utility is once again larger than that of a_1.

For the above experiments, we first compare the cumulative data delivery rate over the affected Link l_2 achieved by REAM and the static planner for two different network loss rates of 0.1 and 0.5. As shown in Figure 5.15, the loss incurred by REAM is significantly lower as compared to that by the static planner. The difference ranges from 6% for 0.1 network loss rate to 27.5% for 0.5 network loss rate. Since this difference in delivery rate is a function of the network loss rate on the affected link, the link quality of the alternative paths, as well as the duration for which the link disruption persists, it would be even bigger for large-scale disruptions on multiple network links.

We then compare the benefit achieved over time by REAM and the static planner during the experiments. We measure the accuracy over time by multiplying the expected accuracy of each selected action plan with the measured message delivery rate. We calculate the delivery rate for each action plan as the number of packets received each timestep divided by the total number of packets sent for that plan. Fig. 5.16 shows the accuracy values per timestamp achieved by both approaches over the entire experiments for both 0.1 and 0.5 network loss rates. We see that before the introduction of the network loss rates, both REAM and the static planner achieve the same high accuracy since they both use the high utility action

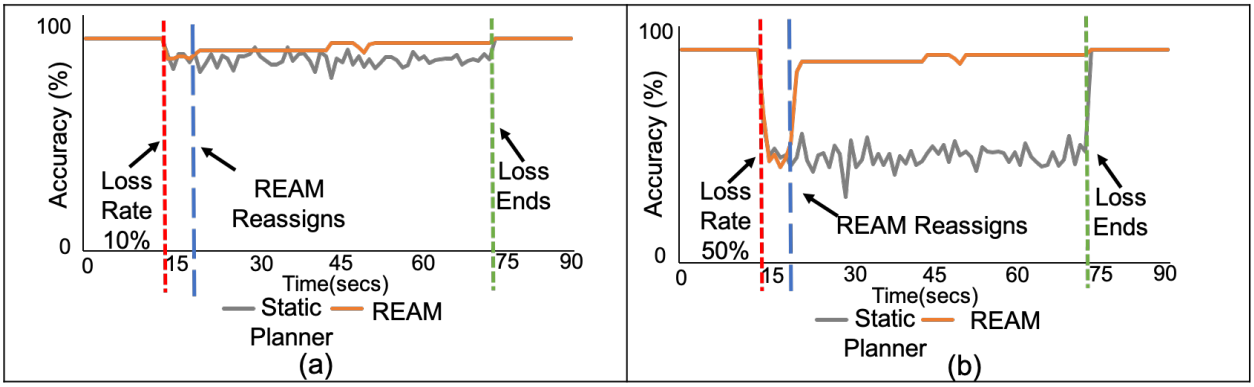


Figure 5.16: Average monitoring accuracy obtained by REAM and the static planner for network loss rates of: (a) 0.1 and (b) 0.5, respectively

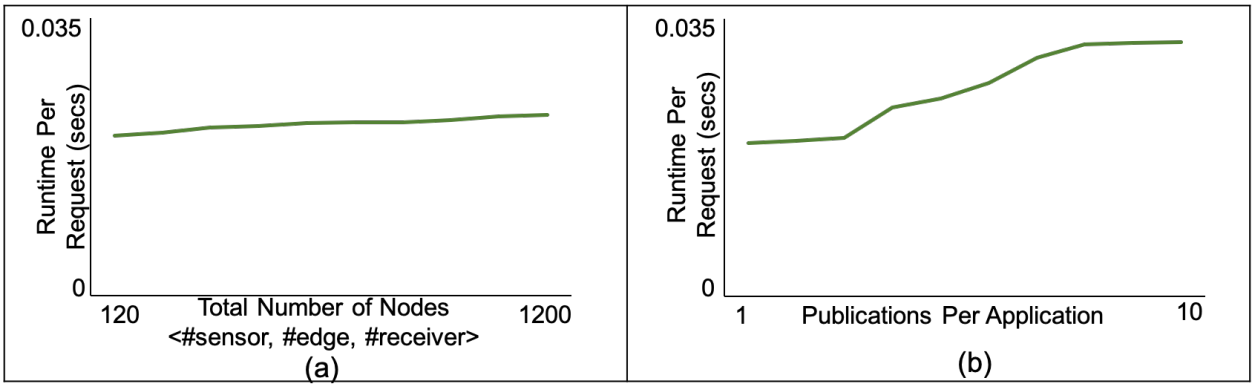


Figure 5.17: Running time taken per application for deployments ranging (a) from 120 to 1200 nodes and (b) from 1 to 10 tasks

plan a_2. We also see that the introduction of network loss results in a drop in accuracy for both approaches. However, REAM's agent recomputes the utility values and selects action plan a_1 unlike the static planner that continues with a_2 and thus suffers from lost packets. The average difference in accuracy levels ranges from 11% with 0.1 network loss rate and 42% with 0.5 network loss rate. When the network condition on the affected link recovers, REAM switches back to action plan a_2.

5.4.5 Evaluating Scalability

In this section, we evaluate the scalability of REAM for large-scale community infrastructure deployments consisting of a multitude of sensors, edge servers, stakeholders or receivers, and monitoring applications with varying action plan complexities. We use the same prototype and simulated input setup as in the network loss experiment. We first varied the total number of sensors, edge servers, and receivers, and then measured the time taken by REAM to handle action plan selections for multiple applications. Figure 5.17(a) shows the time taken per application by REAM for 100 applications, in environments ranging from 120 nodes (100 sensors, 10 edge servers, and 10 receivers), to 1200 nodes (1000 sensors, 100 edge servers, and 100 receivers). Each action plan candidate for the applications was set to have 3 tasks (sensing, compute, and publication). We observed that the time taken per application remained relative constant despite the 10x scaling of the environment size. The runtime of REAM ranged from 0.020 to 0.022 seconds to select and publish action plans for each application which shows a near constant runtime, thus showing REAM's effectiveness in handling large-scale infrastructure deployments.

We then varied the number of tasks (i.e., the number of data processing steps) associated with each action plan from 1 to 10 for 100 applications. We kept the environment size constant at 1200 (1000 sensors, 100 edge servers, 100 receivers). Fig. 5.17(b) shows that there is a small increase in the time taken to select and publish action plans per application while increasing the number of tasks per action plan. The runtime ranges from 0.019 to 0.032 seconds for an average of 0.027 seconds per application. Given how small these runtimes are, it is an insignificant increase in the runtime of REAM when handling a larger number of tasks per action plan.

5.5 Chapter Summary and Discussion

In this chapter, we address the operational challenges of infrastructure monitoring under heterogeneity and varying community environments (structure and behavior). We develop a resource efficient adaptive monitoring framework, REAM, that balances the resource requirements and objectives of multiple infrastructure monitoring applications in order to provide good quality monitoring of community infrastructure, while incurring low compute, networking, and energy costs. REAM allows community stakeholders and users to define monitoring workflows or pipelines of sensing, communication, and analytics that can be leveraged to serve each monitoring application. It leverages reinforcement learning (RL) agents that identify and learn structural and behavioral patterns in the community infrastructure, to determine an optimal policy for selecting different monitoring workflows depending on the given state of the community infrastructure and environment. Experimental results show that REAM achieves high levels of monitoring performance at significantly low resource consumption costs, while adaptively switching between different workflows to handle changing infrastructure conditions.

Chapter 6

Sharing Monitoring Solutions Across Communities

The solutions presented in the previous two chapters addressed deployment and operational challenges for infrastructure monitoring within one community. However, developing monitoring solutions for each individual community requires time and effort, and oftentimes may not be possible due to a lack of sufficient resources like sensor instrumentation, monetary capital, available data and compute resources, etc., that are required to train effective models. Hence, extending single-community solutions to work or generalize across communities can provide significant benefits by enabling communities to collaboratively achieve good quality infrastructure monitoring and save on costs and time by sharing data and models. In this chapter, we present our approach to enable cross-community solutions while addressing challenges of centralized training, privacy, and data biases. We develop a solution for the distributed training of monitoring models that can generalize and be deployed across communities without a loss in performance.

6.1 Chapter Overview

Community infrastructure monitoring applications driven by Internet-of-Things (IoT) devices generate a massive amount of data from sensors or devices at the edge. Machine learning models are trained on these datasets to learn complex patterns and correlations between infrastructure events and the different data features or attributes that are measured. The objective of these models is to identify the underlying causal relationship between the different data features and the target event, and often require large amounts of training data and compute resources to achieve good performance. Since different locations and communities observe a diverse range of events, training models on data shared from these different sources can improve their robustness and event detection capabilities.

Similarly, model sharing between communities is also beneficial, since models that can be reused by different communities can save them the time and costs required to extensively instrument infrastructure and collect data. Moreover, some communities do not have sufficient monetary capital and resources to obtain the data and computation needed to train effective monitoring models. This can result in communities with access to different levels of resources having starkly different monitoring capabilities both in terms of the quality and extent of monitoring. Hence, sharing data and models between communities can help improve the robustness of the monitoring models, save time and costs associated with model training, and ensure that every community irrespective of their resource availability, can obtain good monitoring models for their infrastructure. However, such sharing between communities is challenging today due to issues of centralized training and the inability to handle data biases:

- **Centralized training:** Given the vast quantities of data generated by IoT sensors, centralized data collection and training places enormous strain on transmission network bandwidths and storage requirements needed for other applications. Additionally, a central location acts as a single point of failure that can be compromised by malicious

actors since infrastructure monitoring often results in the collection of personal and sensitive data. For instance, smart building applications collect WiFi usage data, images, and other mobile device information from occupants. On the other hand, critical community infrastructures like water networks store sensitive information like network vulnerabilities and water treatment schedules. Such kinds of information, if obtained by malicious actors, can cause significant adverse impacts on individuals and the community at large. These issues preclude communities and stakeholders from sharing data and information, necessitating distributed training solutions.

- **Data biases:** During the training process, models try to identify causal patterns in the community infrastructure by learning from correlations present between the different features being measured and the target event. However, some of these correlations may be biased or spurious, where they do not exist outside of the collected data. Models relying on these features often perform poorly on new data where the biases are no longer present. This makes it challenging to train and share models using data from one or more communities, since they need to ignore existing community biases that may not exist in other communities in order to perform well.

In this chapter, we address the above challenges and develop an approach to enable the distributed training of infrastructure monitoring models that can be shared or generalized across communities without a loss in performance. Our contributions would enable resource-poor communities to obtain good quality infrastructure monitoring models without needing extensive capital and resources, and also allow each community to improve the robustness of their models by training them on diverse events observed in other communities without the need to share sensitive data. Our key contributions include:

- Develop a solution, titled FedGen, to train generalizable infrastructure monitoring models in a distributed manner that can be shared and re-used effectively across com-

munities.

- Present approaches to leverage the federated learning paradigm for distributed training and to improve generalizability by identifying biases in data.
- Develop a novel masking function that leverages individual feature stability during model training to distinguish between causal and spurious (biased) features in a distributed manner.
- Formal proof of the correctness of our approach in learning only the causal features and ignoring biases in the data.
- Conduct extensive evaluation of FedGen on real-world datasets from different infrastructure monitoring applications and comparisons with existing state-of-the-art approaches. Our results demonstrate the significant improvement in generalizability achieved by models trained using FedGen, while overcoming the drawbacks of centralized training.

6.2 Distributed Training of Generalizable Monitoring Models

Traditionally, machine learning models for monitoring community infrastructure are deployed on central cloud servers and trained on data collected from distributed sensor deployments in communities. However, the increased popularity of IoT-driven monitoring has resulted in the generation of a massive amount of data from these distributed deployments. The vast quantities of data, coupled with network bandwidth limitations, and privacy concerns have made it impractical to gather all the data from these sensors to a single server to conduct centralized training.

To address the need for a distributed training approach for infrastructure monitoring applications, federated learning has emerged as an attractive paradigm to allow local devices or clients to collaboratively train a shared global model. The typical federated learning paradigm involves two stages – (i) clients train models with their local datasets independently, and (ii) a central server gathers the locally trained models and aggregates them to obtain a shared global model. Federated learning is also privacy-aware, since instead of sharing sensitive raw data, only individual model weights or parameters need to be shared between the clients and the central server. For infrastructure monitoring, the clients can correspond to data from individual devices, local servers, specific locations, or even entire communities themselves. This semantic flexibility in federated learning therefore enables different infrastructure monitoring applications to define clients and local models depending on their objectives.

A standard approach for model aggregation in federated learning has been FedAvg [96], where parameters of local models are averaged element-wise with weights proportional to sizes of the client datasets. While most existing work using federated learning for smart community applications use FedAvg, it has been shown to have several shortcomings [86, 171]. Of particular importance, is its poor performance when the data across clients are not independent and identically distributed (i.e., non-iid data). However, the presence of non-iid data for training is often the case with infrastructure monitoring applications, where sensors and devices can have differing amounts of data, and the data distributions can also vary across devices due to differences in sampling rates and device heterogeneity, among other reasons.

Additionally, infrastructure monitoring data often contains biases, which are also not addressed by existing federated learning approaches. The goal of every monitoring model is to capture the underlying causal relationship or **invariant correlations** between different features or attributes and the target event. However, data collected from infrastructure

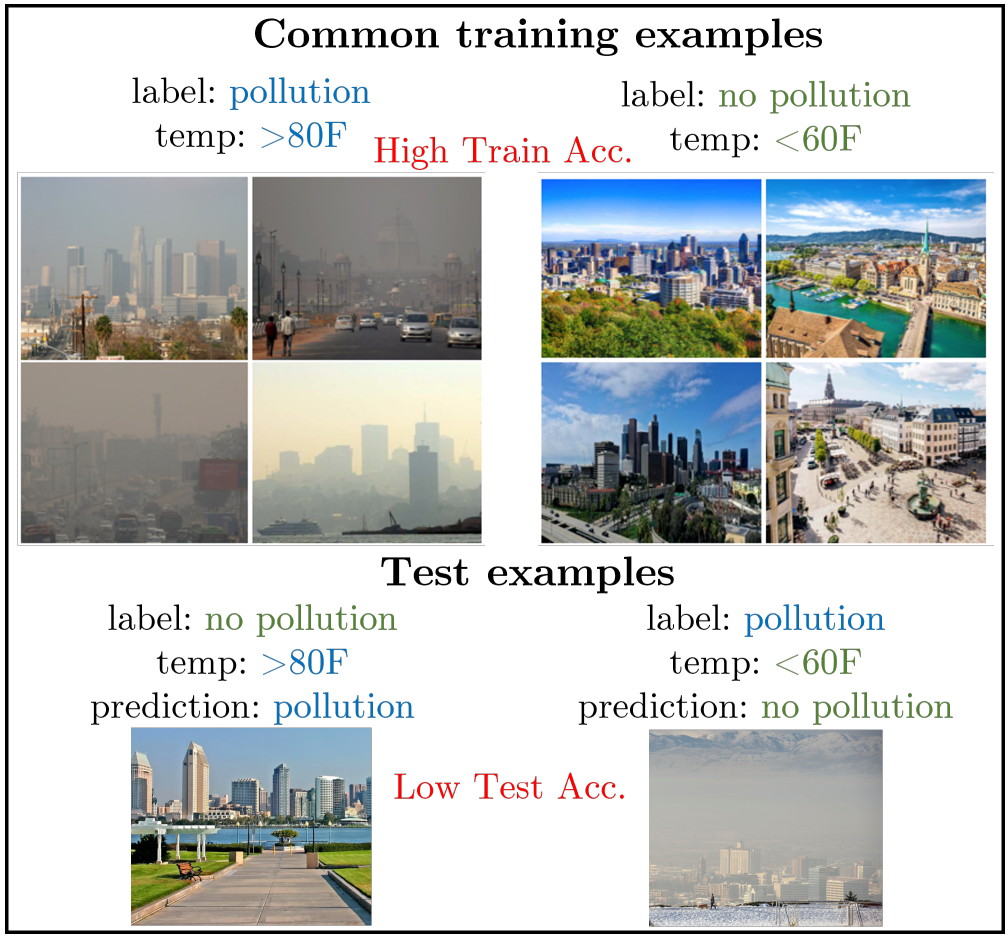


Figure 6.1: Spurious correlation between event (air pollution) and temperature only holds in training data

in a specific community, can often have inherent biases or **spurious correlations** present between some features and the target event. These spurious correlations can occur due to sensor locations, sampling issues, and other community-centric characteristics like weather patterns among others. These correlations have no causal relationship with the event outside of the collected data, but because models leverage all correlations during the training process, they also learn and rely on these biases or spurious relationships. This results in models showing poor performance on new data without these biases, from either the same community or other communities, preventing their sharing across communities.

An example of this can be seen in Figure 6.1, depicting images from different cities to detect the presence of air pollution. Additionally, feature measurements like PM2.5 (particulate

matter), temperature, and humidity were also collected. In this example, the sole **invariant correlation** between the target event (i.e., air pollution) and the features is due to high PM2.5 sensor readings. That is, this relationship between air pollution and high PM2.5 measurements would hold true, no matter the community. However, this example also contains a **spurious correlation** between the event and the temperature readings, where most places with air pollution have high temperatures ($> 80F$), and most places with no air pollution have low temperatures ($< 60F$). This correlation is captured by the model during training, leading it to believe that communities with high temperatures would likely show air pollution, while cold communities would not have air pollution. However, this bias would obviously not hold outside of the collected data as shown in the test examples in Figure 6.1, where communities with high temperature can have no air pollution and vice-versa. Hence, models influenced by spurious correlations would perform poorly when deployed in other communities where the biases no longer hold.

Improving the generalizability of machine learning models by training them to distinguish between spurious and invariant features is an important challenge that has been predominantly studied in the context of computer vision problems [7, 77]. However, this is also critical to address for smart community and infrastructure monitoring applications, where data biases are prevalent and hard to manually identify. In the rest of this chapter, we present relevant background and our approach to train generalizable models in a federated privacy-aware manner.

6.3 Background

In this section, we provide an overview of federated learning and machine learning generalization. We present formulations and discuss prior work in both domains.

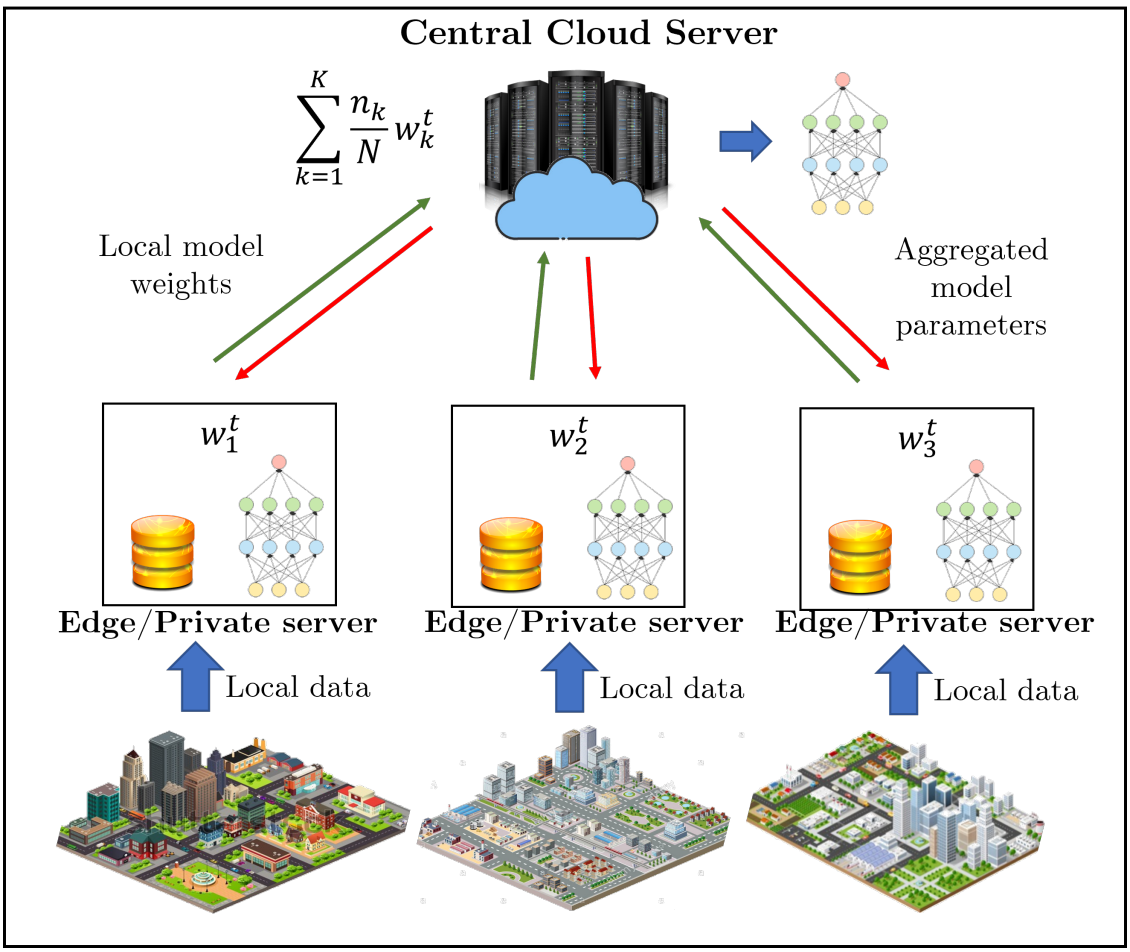


Figure 6.2: Illustration of the FedAvg approach using data from three communities

6.3.1 Federated Learning

Recall that federated learning involves the distributed training of multiple local models or clients and subsequent centralized model aggregation. For smart community settings, these clients could refer to models analyzing data from individual devices or even large sensor deployments across an entire community. We assume that these models are deployed on edge servers, and the centralized aggregated model is present either at the edge or on the cloud. Figure 6.2 illustrates the general federated learning paradigm, where we depict clients as individual communities with their own local models and data. These local model parameters are periodically transmitted to the central server, which aggregates the parameters and returns them back to update the local models.

Consider a set of K clients, each with data samples drawn from $\mathcal{D}$ distributions or domains on $X \times Y$, where X is the set of input features and Y is the target label. We denote n_k as the number of data samples on client $k \in K$, and $N = \sum_{k=1}^{K} n_k$ as the total number of samples across the K clients.

For the federated learning setting, we define the local loss for the model on client k, which maps the model parameters w_k to the expected loss on the local data distribution $D_k \in \mathcal{D}$ for a given loss function ℓ as:

$$f_k(w_k) = \mathbb{E}_{(x_k, y_k) \sim D_k}[\ell(x_k, y_k); w_k] \tag{6.1}$$

The central server aggregates these local models into a central model using the following central objective function:

$$F(w) = \sum_{k=1}^{K} \frac{n_k}{N} f_k(w_k) \tag{6.2}$$

where w is the parameters of the aggregated central model. The overall goal of the federated learning paradigm is to find model parameters w^* such that:

$$w^* = \arg\min F(w) \tag{6.3}$$

Federated Averaging (FedAvg) proposed by McMahan et al. [96] has emerged as the de-facto optimization approach for smart community applications in the federated setting. At each iteration of the algorithm, FedAvg performs E local epochs of stochastic gradient descent (SGD) on a random subset of clients. The clients then communicate their local model updates to the central server, where they are averaged as shown in Figure 6.2. The details of FedAvg are summarized in Algorithm 5.

Algorithm 5 Federated Averaging (FedAvg)

Input: Devices $k \in K$, local epochs E, learning rate η, global model w randomly initialized
for $t = 1 \rightarrow T$ **do**
    Server selects subset K_t of K devices at random
    Server sends w^t to devices in K_t
    for each client $k \in K_t$ in parallel **do**
        Update w^t for E epochs of SGD to get w_k^{t+1}
        Send w_k^{t+1} back to the server
    Server aggregates $w^{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} w_k^{t+1}$

However, it has been shown that when the local clients have non-iid data partitions and heterogeneous local objectives f_k, the local training in FedAvg may lead each device to optimize its local objective as opposed to the central objective – potentially hurting convergence or even causing divergence of the central model due to the underlying heterogeneous data distributions [20]. In order to improve the convergence under non-iid settings and to reduce the total number of communication rounds (by setting high local epochs), several alternative approaches have been proposed [100, 171]. For example, Li et al. [86] proposed FedProx, which adds a proximal term to the local client's update functions, thereby limiting the impact of local updates by keeping them close to the global objective. In particular, they define the following objective:

$$w_k^{t+1} \approx \min_{w_k} h_k(w_k, w^t) = f_k(w_k) + \frac{\mu}{2}||w_k - w^t||^2 \tag{6.4}$$

where μ is a hyper-parameter and we can see that when $\mu = 0$, this reduces to FedAvg. However, these approaches only focus on improving convergence in non-iid settings, but do not address the additional challenge of data biases and spurious correlations which can cause training distributions to differ significantly from test distributions. Thus, as we will demonstrate in our experimental evaluation, these approaches do not result in robust and generalizable models.

6.3.2 Generalization of Monitoring Models

Machine learning models have always been evaluated by their performance on unseen test data. Classical machine learning approaches assume that the unseen test data are drawn i.i.d from the same data distribution as the training set used to train the model. However, in many real-world applications including infrastructure monitoring, this is often not the case. When this discrepancy occurs, classical training approaches often result in models failing to perform on different data distributions. Recent efforts have provided evidence that models trained with the iid assumption rely on statistically informative but non-causal features in the data (i.e.) spurious correlations [11, 51, 63]. These spurious correlations are often misleading features/attributes that hold for a majority of training examples but will not always hold. Hence, models that learn and rely on these spurious correlations would demonstrate good training accuracy, but perform poorly on new test data where they no longer hold. This has prompted the need for solutions that can train generalizable models that are robust to spurious correlations.

Generalization is a harder problem to address than the related areas of domain adaptation and meta-learning, which have been studied in the context of community monitoring applications. These other areas assume that the test data distribution is known apriori, and often some unlabeled test data is also available. However, in generalization, no such assumptions are made, and the test distributions are unknown. Thus far, efforts to develop solutions for generalization have looked at centralized training, and not distributed training settings.

As described in Section 6.2, generalizable monitoring models need to be able to distinguish between spurious and invariant features, thereby ignoring spurious correlations present in the training data. Given the set of input features X used by the client models, the set of invariant features X^I across all clients is one where the event prediction probability is consistent across all data distributions, (i.e.) $p(Y|X_i \in X^I, \mathcal{D})$ is approximately constant. Conversely, the

set of spurious features X^S consists of features whose prediction probabilities differ across distributions due to the presence of data biases. Hence, it follows that $X^I \cup X^S = X$, and $X^I \cap X^S = \emptyset$, (i.e.) a feature cannot be both invariant and spurious.

The classical centralized approach to training machine learning models uses Empirical Risk Minimization (ERM), which tries to minimize the average loss over all training examples in a distribution agnostic manner:

$$F(w) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\ell(X, Y); w] \tag{6.5}$$

where $\mathcal{D}$ refers to the different distributions across the K clients. While ERM has been shown to work well in practice for i.i.d. data [157], it often fails when test distributions differ significantly from training distributions [153].

To overcome this, several approaches have been proposed, including the popular Invariant Risk Minimization (IRM), proposed by Arjovsky et al. [7]. IRM searches for an invariant representation of input features across the different distributions. We paraphrase the IRM principle as : *An invariant representation* $\Phi(X)$ *is one such that the optimal linear predictor* w *is the same across all distributions* $D \in \mathcal{D}$. They show that finding the invariant predictor, $w \circ \Phi$, requires solving the following bi-level optimization problem:

$$\min_{\Phi, w} \sum_{D\in\mathcal{D}} F(w^\top \Phi(X, D)) \tag{6.6}$$

$$\text{s.t. } w \in \underset{\bar{w}}{\text{argmin}} \;\; F(\bar{w}^\top \Phi(X, D)), \;\; \forall D \in \mathcal{D} \tag{6.7}$$

However, since this optimization is highly intractable, particularly when Φ is non-linear,

they propose a tractable variant (IRMv1) :

$$\min_{\Phi} \sum_{D \in \mathcal{D}} F(\Phi(X, D)) + \lambda \|\nabla_w F(w^\top \Phi(X, D)\|_2^2 \tag{6.8}$$

where $\lambda \in [0, \infty)$ is a regularizer that balances between predictive power within a distribution (ERM), and the invariance of the predictor across distributions. There have been several extensions to the IRM framework. For instance Ahuja et al. [2] propose a game theoretic approach to IRM, while Krueger et al. [77] introduce the notion of risk extrapolation to encourage strict equality between training losses.

However, these IRM-based approaches suffer from two inherent weaknesses. First, they rely on the assumption that the different training distributions $\mathcal{D}$ are known apriori. Second, they require perfect segmentation of the training data into these distributions. In practice however, since the data at a single client could consist of multiple distributions, it is challenging to identify and distinguish data from individual distributions, resulting in imperfect segmentation of data.

To demonstrate the sensitivity of IRM to imperfect data segmentation, we use the Punctuated SST-2 dataset [34]. It consists of sentences and their binary sentiment labels (positive or negative) divided into two training distributions. A punctuation mark, either a '!' or '.', is introduced as a spurious feature with an 80% and 90% correlation with each of the binary sentiment labels in the two training distributions respectively, and only has a 10% correlation in the test distribution. Any model influenced by the punctuation feature rather than the sentence while predicting the sentiment, would do well during training but perform poorly at test time. To simulate imperfect segmentation of data into the training distributions, we "incorrectly" assign a percentage of examples from the first distribution to the second.

Figure 6.3 shows the resulting out-of-distribution accuracy on the test distribution by the

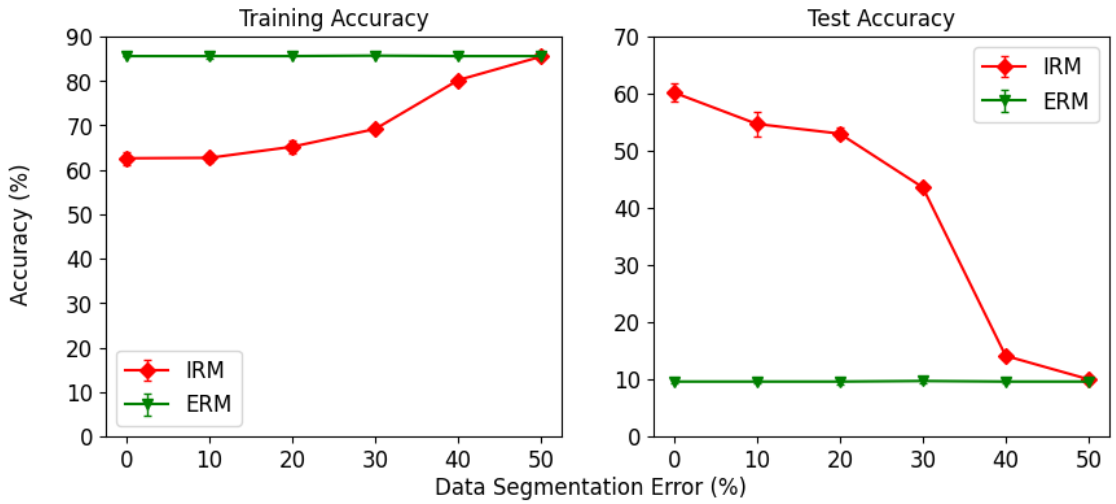


Figure 6.3: Comparing the sensitivity of Invariant Risk Minimization (IRM) to Empirical Risk Minimization (ERM) for imperfect segmentation of data into distributions. The test accuracy of IRM degenerates to that of ERM when the training data is not segmented correctly, resulting in no model generalization.

IRM model as compared to a standard Empirical Risk Minimization (ERM) model where the ERM model tries to minimize the average loss over all training examples. We observe that with perfect data segmentation (0% error), the IRM model is not influenced by the spurious feature correlation and achieves good generalization unlike the ERM model. However, as the segmentation error increases, its accuracy drops significantly. The IRM model becomes heavily influenced by the spurious punctuation feature, as evidenced by the high training accuracy and low out-of-distribution test accuracy. It degenerates to the accuracy obtained by ERM when there is no difference in the spurious correlations between the two segmented distributions, thus achieving poor generalization.

The example above highlights the drawbacks of IRM-based approaches and motivates the need for a distribution-agnostic solution for the generalization of machine learning models.

6.4 FedGen: Generalizable Federated Learning

We have described the shortcomings of existing federated learning approaches being unable to handle biases and spurious correlations, and generalization approaches relying on centralized training and not being distribution-agnostic. In this section, we address all these shortcomings by developing a generalizable federated learning approach, named FedGen, which enables models to identify spurious and invariant features during distributed training.

Our approach to determining whether the i^{th} feature $X_i \in X$ is spurious or invariant, is by measuring the stability of its parameter weight w_i. Recent work [66, 163] has shown that if X_i is a causal or invariant feature, w_i converges to a fixed magnitude, (i.e.) $\mathbb{E}[Y|X_i] = c$ for some constant value c, across all training iterations. Whereas if $\mathbb{E}[Y|X_i]$ is changing, w_i would keep changing as well, and hence spurious features have parameter weights that exhibit high variance. This definition is equivalent to learning features whose correlations with the target variable are stable.

We leverage this intuition for our federated setting and define a set of masks $M_k = \{m_{1k}, ..., m_{nk}\}$ over the input features for each of the $k \in K$ clients, where $m_{ik} \in \mathbb{R}$ and n is the number of input features in X. We update the masks during training by using the variances in the feature weights to emphasize invariant features and suppress spurious ones. During each training epoch, we update the local masks for each client $k \in K$ as:

$$m_{ik} \leftarrow m_{ik} - \alpha(v(w_{ik})) + \frac{1}{n}\sum_{i=1}^{n} v(w_{ik}),\ \forall m_{ik} \in M_k \tag{6.9}$$

where hyper-parameter α serves as a scaling factor, $v(w_{ik})$ is the variance of the weights of feature X_i on client k, and the last term is the average variance of all feature weights. We know from our earlier intuition that the variance of the weights of invariant features is low and that of spurious features would be high. Hence, we see that updating the local masks on each

client using Equation 6.9 results in the masks of invariant features gaining in value on each of the K clients since their variance is lower than the average (i.e.) $\frac{1}{n}\sum_{i=1}^{n} v(w_{ik}) - \alpha(v(w_{ik})) > 0$. Masks of spurious features on the other hand, become progressively negative, since the variance of their weights coupled with α is larger than the average which is brought down by invariant features. Since our update function does not enforce a bound on the mask values (i.e.) $m_{ik} \in [-\infty, \infty]$, multiplying the masks with each input feature can distort the scale of the feature values. In order to retain the scale of the original feature values, we multiply the masks with the sigmoid function σ which is bounded between $[0, 1]$. Hence using the masks and the update function on the input features for each local client results in:

$$\sigma(m_{ik}) \odot X_i \rightarrow \begin{cases} X_i & \text{if } X_i \text{ is invariant} \\ 0 & \text{if } X_i \text{ is spurious} \end{cases} \quad , \forall X_i \in X \tag{6.10}$$

where $\odot$ denotes element-wise multiplication. The local loss on each client for the federated setting described in Equation 6.1 can be represented for FedGen as:

$$w_k^* = \min_{w_k} \underbrace{f_k'(w_k)}_{\text{local loss}} + \underbrace{\lambda \|\nabla_{w_k} w_k^\top f_k'(w_k)\|_2^2}_{\text{FedGen penalty}}, \text{ where} \tag{6.11}$$

$$f_k'(w_k) = \mathbb{E}_{(x_k, y_k) \sim D_k}[l(\sigma(M_k) \odot x_k, y_k); w_k]$$

The penalty term here is inspired by IRM and serves as a regularizer for the local models that penalizes doing too well on one data distribution (i.e., possibly relying on spurious features) and rewards doing well across distributions (i.e., relying on invariant features). However, note that FedGen does not require any form of data segmentation or prior knowledge about distributions. FedGen then trains the centralized model by communicating and subsequently aggregating both the local model parameters as well as the local masks of the clients. We

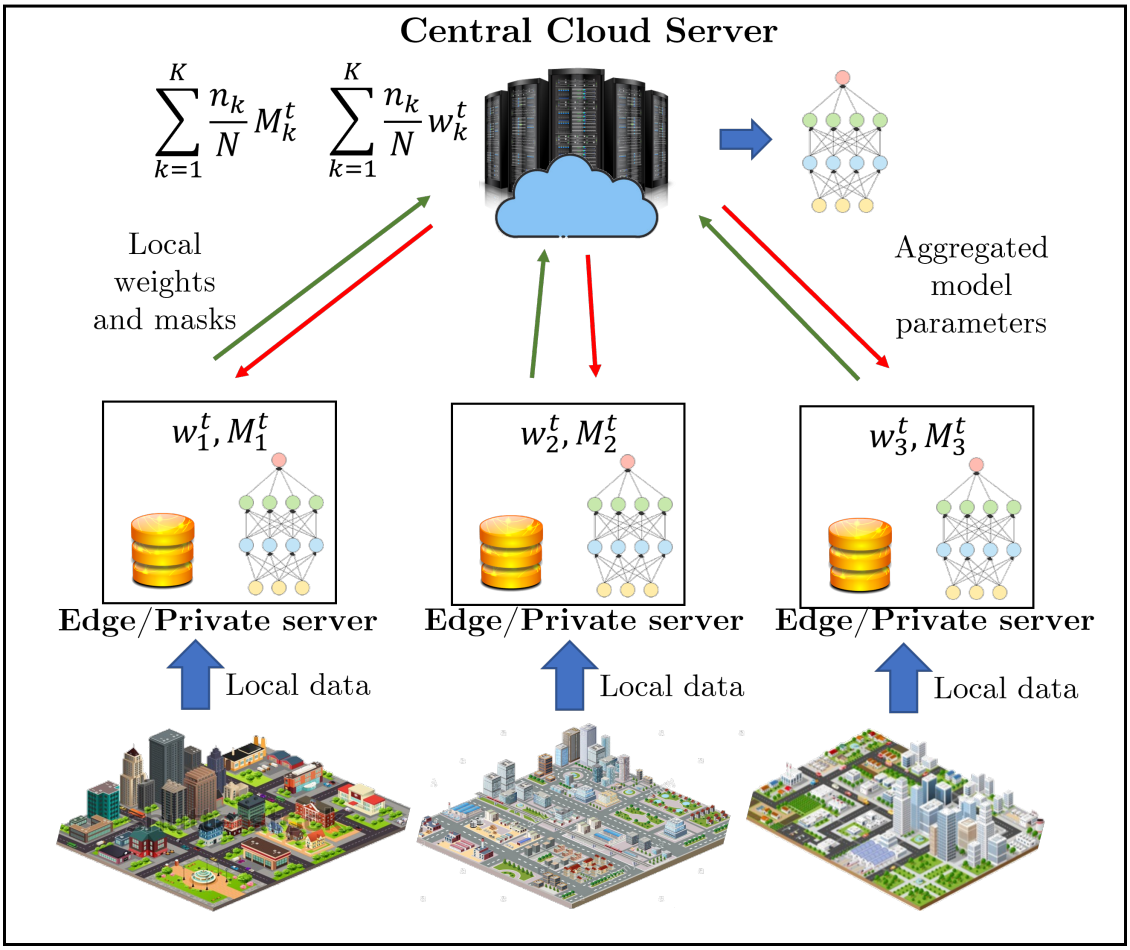


Figure 6.4: Illustration of our proposed FedGen approach

use the same element-wise averaging for aggregation as used by prior work:

$$F(w) = \sum_{k=1}^{K} \frac{n_k}{N} f_k'(w_k) \ , \quad \mathbf{M} = \sum_{k=1}^{K} \frac{n_k}{N} M_k(w_k) \tag{6.12}$$

where F and $\mathbf{M}$ are the globally aggregated model parameters and masks respectively. Aggregating the masks allows us to build a consensus between the clients, where features that are invariant across all clients will be further emphasized, while features that are deemed spurious by clients will be suppressed. Our federated generalization approach is described in Algorithm 6 and illustrated in Figure 6.4. We next formally prove that FedGen results in an aggregated model that is generalizable in Theorem 6.1 by formulating it as a minimax

Algorithm 6 Federated Generalization (FedGen)

Input: Devices $k \in K$, local epochs E, learning rate η, global model w randomly initialized
for $t = 1 \rightarrow T$ **do**
    Server selects subset K_t of K devices at random
    Server sends w^t to devices in K_t
ClientUpdate:
Initialize masks $m_{ik} = 1,\ \forall i : 1 \rightarrow n$
for $e = 1 \rightarrow E$ **do**
    Compute $\ell_{loc} = [\ell(\sigma(M_k) \odot x_k, y_k); w_k]$
    Compute $\ell_1 = \|w_k\|_1$ ▷ $L1$ regularization
    Compute $\ell_{pen} = \lambda \|\nabla_{w_k} w_k^\top f_k'(w_k)\|_2^2$
    $\mathcal{L} = \ell_{loc} + \ell_1 + \ell_{pen}$ ▷ Total loss
    $w_k = w_k - \eta \nabla \mathcal{L}$
    m_{ik} += $\frac{1}{n} \sum_{i=1}^{N} v(w_{ik}) - \alpha(v(w_{ik})), \forall m_{ik}$
Server aggregates $w^{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} w_k^{t+1}$
Server aggregates $M^{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} M_k^{t+1}$

problem and showing that the resulting model minimizes the loss using invariant features, even under the most adverse unseen test distributions.

Theorem 6.1. *Given training distributions D^{tr} and a test distribution D^{te}, the set of invariant features X^I is the saddle point of the following minimax problem:*

$$X^I = \min_{w} \max_{X^I, X^S} \mathcal{L}_{test}(F(\sigma(\boldsymbol{M}) \odot X, w); D^{te})$$

where $\mathcal{L}_{test}$ is the cross-entropy loss for the test distribution, $F(w)$ is the FedGen central objective function, and X^I, X^S denote the set of invariant and spurious features respectively such that $X^I \cup X^S = X$, and $X^I \cap X^S = \emptyset$ (i.e.) they are disjoint.

Proof. For notational simplicity, we denote $\mathbf{Z} = \sigma(\mathbf{M}) \odot X$. For every $\mathbf{Z}$, we can partition it into invariant variables $\mathbf{Z}^I$ and non-invariant variables $\mathbf{Z}^S$ as:

$$\mathbf{Z}^I = \sigma(\mathbf{M}) \odot X^I\ , \quad \mathbf{Z}^S = \sigma(\mathbf{M}) \odot X^S. \tag{6.13}$$

Consider a test distribution D^{*te}, where the set of spurious features X^{S*} are not predictive of the output Y, and only the invariant features X^I are predictive of Y, (i.e.)

$$p(Y|\mathbf{Z}, D^{*te}) = p(Y|\mathbf{Z}^I, D^{*te}), p(Y|\mathbf{Z}, D^{tr}) = p(Y|\mathbf{Z}^I, D^{tr}) \tag{6.14}$$

Therefore, $\mathcal{L}_{test}(F(\mathbf{Z}, w)\ ; D^{*te})$

$$\begin{aligned}
&= H(p(Y|\mathbf{Z}, D^{*te}); p(Y|\mathbf{Z}, D^{tr})) \\
&\overset{(i)}{=} H(p(Y|\mathbf{Z}^I, D^{*te}); p(Y|\mathbf{Z}^I, D^{tr})) \\
&\overset{(ii)}{=} H(p(Y|\sigma(\mathbf{M}) \odot X^I, D^{*te}); p(Y|\sigma(\mathbf{M}) \odot X^I, D^{tr})) \\
&\overset{(iii)}{=} H(p(Y|X^I, D^{*te}); p(Y|X^I, D^{tr})) \\
&= \mathcal{L}_{test}(F(X^I, w)\ ; D^{*te})
\end{aligned} \tag{6.15}$$

where $H(\cdot)$ is the cross-entropy loss function. Step (i) is obtained from applying equation (6.14). Step (ii) is obtained by applying equation (6.13), and step (iii) is due to the property of the masks from equation (6.10).

Recall that X^{S*} was assumed to be non-predictive of Y. However, in most cases, the spurious feature X^S would have some predictive power over Y in the training environment. Hence, from the definition of spurious features X^S, their biased influence on the model performance during training will lead to an increased loss in the worst case test environment:

$$\max_{X^S} \mathcal{L}_{test}(F(\mathbf{Z}, w); D^{te}) \geq \mathcal{L}_{test}(F(\mathbf{Z}, w); D^{*te}) \tag{6.16}$$

Recall X^I denotes the set of invariant features, thus $p(Y|X^I, D^{test})$ does not depend on X^S.

Therefore,

$$\max_{X^S} \mathcal{L}_{test}(F(X^I, w); D^{te}) = \mathcal{L}_{test}(F(X^I, w); D^{*te}) \tag{6.17}$$

By combining equations (6.15), (6.16), and (6.17), we have:

$$\max_{X^S} \mathcal{L}_{test}(F(\mathbf{Z}, w); D^{te}) \geq \max_{X^S} \mathcal{L}_{test}(F(X^I, w); D^{te}) \tag{6.18}$$

The above formulation holds for all X^I. Hence, taking the maximum over X^I in equation (6.18) preserves the inequality,

$$\max_{X^I, X^S} \mathcal{L}_{test}(F(\mathbf{Z}, w); D^{te}) \geq \max_{X^I, X^S} \mathcal{L}_{test}(F(X^I, w); D^{te})$$

which in turn implies,

$$X^I = \min_{w} \max_{X^I, X^S} \mathcal{L}_{test}(F(\mathbf{Z}, w); D^{te})$$

□

6.5 Experiments

In this section, we present an extensive evaluation of FedGen on real-world infrastructure monitoring applications and compare its performance with other centralized and federated learning approaches. Similar to prior work [7, 34, 2], we augment these datasets with spurious correlations to measure the generalizability of the resulting models from the different approaches. This spurious correlation is strongly present in the training data distributions D^{tr} between the target events Y and a feature X^S (i.e., $p(Y|X^S, D^{tr} \geq 0.8)$, and this corre-

Dataset	#Clients	Train Samples	Test Samples
HAR	30	7352	2947
Stormwater	15	132,227	14,237
Air Quality	61	549,162	38,398

Table 6.1: Summary of Federated Datasets

lation does not hold in the test data distributions (i.e., $p(Y|X^S, D^{te}) \leq 0.1$).

We assess the quality of generalization of the aggregated model resulting from the different approaches as the classification accuracy obtained on the unseen test distribution without spurious correlations, disjoint from the set of training distributions. We also conduct additional experiments to compare the impact of local training epochs on accuracy, convergence rates achieved, and an ablation study to measure the contribution of the different components of our proposed masking function.

6.5.1 Datasets

We use three real-world datasets depicting monitoring applications across stormwater infrastructure, personal wearables, and air quality. Table 6.1 shows a summary of these different datasets.

Stormwater Contamination Detection

We obtain stormwater quality data from 15 different cities across USA [46] along with their corresponding stormwater contamination event information. We treat the data from each city as a client for our federated learning setting. The data comprises of chemical sensor readings such as turbidity, conductivity, pH, temperature, DO2, in addition to environmental attributes like humidity and precipitation, and the model's objective is to identify the occurrence of stormwater contamination given these different measurements. In this application,

the chemical sensors are the invariant features for the occurrence of contamination. However, we divide the cities into training (12 cities) and test (3 cities) datasets, such that there is a strong spurious correlation between contamination events and high levels of humidity for cities in the training set, which does not hold for those in the test set.

Human Activity Recognition (HAR)

The HAR dataset [6] consists of smartphone accelerometer and gyroscope readings recorded from 30 individuals performing six activities (walking, standing, sitting, etc.). The model's objective is to predict the activity being performed given the sensor readings. The data consists of sequences of 128 time-steps of sensor readings which correspond to any particular activity. We consider the data from each individual as a local client, and augment a strong spurious correlation between the specific activity performed and the smartphone model used by individuals in the training set (25 individuals), which does not hold for the individuals in the test set (5 individuals).

Air quality monitoring

The dataset [182, 46, 67] consists of air quality sensor measurements (PM2.5, PM10, NO_2, O_3, etc.) from 61 cities across USA, China, India, and Europe, along with meteorological data such as temperature, precipitation and wind speed. The target event reflects the air quality index (AQI) [3] consisting of six classes ranging from good to hazardous, and the model's objective is to classify the sensor and meteorological measurements into the appropriate category. The invariant features for identifying AQI are the air quality sensors, however we divide the clients such that there is a strong spurious correlation between the temperature values and the different AQI classes in the training set (45 cities), that does not hold for those in the test set (16 cities).

6.5.2 Benchmarks

We compare FedGen to four benchmark approaches:

- ERM: Standard empirical risk minimization approach that minimizes the average loss over the entire training data. Used to compare performance with centralized training.
- FedAvg: Federated averaging approach by McMahan et al. [96] where model parameters are averaged element-wise.
- FedProx: Approach by Li et al. [86] to handle heterogeneous non-iid data in federated settings using a proximal term to keep local models close to the global model.
- Inv-FedAvg: FedAvg trained on data without any spurious features. This approach reflects the setting where spurious correlations are not present.

6.5.3 Implementation Details

We implement FedGen, ERM, FedAvg, and Inv-FedAvg using the PyTorch library [108] and implement FedProx based on their publicly available code[1]. For each dataset, we tried both MLP and LSTM classifiers of different architectures and selected the model that performed the best. We used the same model architecture for all comparison approaches to ensure a fair comparison. We used the cross-entropy loss for classification during training and the Adam optimizer. We searched for hyper-parameters based on the values in Table 6.2, and chose the configurations with the best performance for each of the approaches. We also assume the central server samples all the local clients to join the training process in every communication round for simplicity.

[1]https://github.com/litian96/FedProx

Hyper-parameter	Range
Learning rate	$[10^{-5}, 10^{-1}]$
Regularization weight	$[10^{-6}, 10^{-2}]$
Mask scaling factor	$[10^{-2}, 10]$
FedProx μ	$[10^{-5}, 10^{-1}]$

Table 6.2: Hyper-parameter ranges tried for all comparison approaches

Algorithm	HAR		Stormwater		Air Quality	
	Train	Test	Train	Test	Train	Test
ERM	98.10	73.22	88.76	65.39	83.17	60.44
FedAvg	98.03	66.38	91.30	63.84	87.49	54.95
FedProx	96.76	70.17	86.74	67.14	82.96	59.28
FedGen (ours)	94.23	**87.39**	75.47	**90.95**	78.44	**82.06**
Inv-FedAvg	93.01	89.97	74.56	93.18	78.13	85.64

Table 6.3: Accuracy achieved by comparison approaches for all datasets

6.5.4 Results

Model accuracy

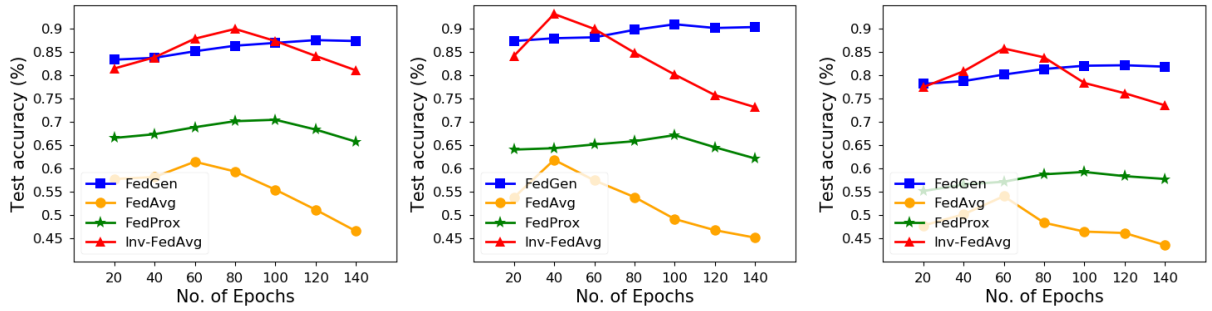
We first measure the accuracy of the final aggregated global model resulting from all the comparison approaches on the test set to compare their generalizability. We identify the final models when they have either converged, started to diverge, or run a sufficient number of rounds (e.g., 200 rounds), whichever comes earlier. We consider the models to converge when the loss difference between two consecutive communication rounds $|F^t - F^{t-1}| < 10^{-4}$, and consider the models to diverge when we observe $|F^t - F^{t-10}| > 1$, similar to definitions by prior work [86, 128].

Table 6.3 summarizes the results, where we see that FedGen outperforms both centralized and federated approaches across all three datasets achieving nearly 23% improvement in accuracy on average. This reflects the significantly increased generalizability of the resulting model from FedGen. We observe that the models from the other approaches rely on the

spurious correlations since their training accuracy across the datasets is high. However, this results in low generalization on the test set where the spurious correlation no longer holds. FedGen, on the other hand, ignores the spurious correlations, and performs nearly as well as Inv-FedAvg, which reflects accuracy when spurious correlations are not present. Additionally, the centralized training approach (ERM) outperforms FedAvg and FedProx, which reflects the inherent drawback of distributed training. FedProx outperforms FedAvg across the datasets, showing that the proximal term helps a little with limiting the influence of spurious correlations.

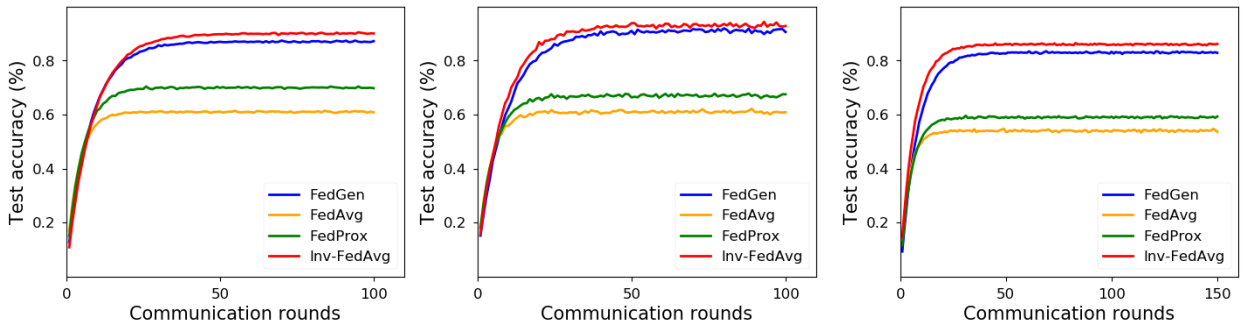
Effect of local training epochs

For federated learning approaches, while a large number of local training epochs E can help in reducing the communication costs, previous work has also shown that the value of E can impact the performance of FedAvg and can sometimes lead to divergence [96, 26, 128]. We conduct an experimental study on the effect of E over the different comparison approaches across all three datasets. The candidate local epochs we consider are $E \in \{20, 40, 60, 80, 100, 120, 140\}$. We run each approach till it either achieves convergence, or the number of rounds exceeds 150 and report the test accuracy achieved in Figure 6.5. We observe that the performance of FedAvg and Inv-FedAvg deteriorates with longer local training, thus showcasing its sensitivity to hyperparameter settings, which matches prior observations made in literature. FedProx only partially alleviates this problem and also demonstrates a drop in accuracy for larger number of epochs. On the other hand, we observe that FedGen benefits from longer training, since it is not influenced by spurious features and can further emphasize invariant features, suggesting that FedGen is the only approach that can be used by local clients to train their models for larger number of epochs without being influenced by local data distributions.



(a) HAR (b) Stormwater (c) Air Quality

Figure 6.5: Effect of number of local training epochs on federated learning approaches



(a) HAR (b) Stormwater (c) Air Quality

Figure 6.6: Convergence rates of different federated learning approaches across all datasets

Convergence Rate

In this experiment we compare the global model convergence rates achieved by the different federated learning approaches. We use the same definition of convergence and tune the number of local training epochs as described in our previous experiments. We report the convergence rate for E that yields the best final aggregated model accuracy over the test set for the different approaches. Figure 6.6 shows the comparison of the convergence rates of the federated learning approaches across the three datasets. We observe that the convergence achieved by the approaches are quite similar, and that FedProx converges the quickest, followed by FedGen and then FedAvg.

Algorithm	Test Accuracy (%)		
	HAR	Stormwater	Air Quality
FedGen	87.39	90.95	82.06
− scaling (α)	74.93	85.51	71.22
− $\sigma(\mathbf{M})$	62.16	63.46	40.01
− penalty term	70.41	81.38	69.38

Table 6.4: Ablation Study

Ablation Study

We perform an ablation experiment to measure the contribution of the components of our masking function and the FedGen penalty term as shown in Table 6.4. We first remove the hyperparameter α used to scale the masks, and observe that there is a drop in accuracy across all three datasets since the degree of spuriousness may not be fully captured, and hence the local models can get influenced by spurious features. We next remove the masking function $\sigma(\mathbf{M})$, essentially reducing the loss function to that used by FedAvg. This is reflected by the accuracy values, which are similar to those achieved by FedAvg, thereby demonstrating that our proposed masking function is the primary reason for model generalization. We finally remove the penalty term while optimizing model weights (Equation 6.11) and again observe a drop in accuracy across all datasets, thereby demonstrating the importance of all components of our proposed masking function.

6.6 Chapter Summary and Discussion

In this chapter, we present our solution, FedGen, for training generalizable monitoring models in a distributed manner that can be deployed across communities. This results in the significant benefits of enabling communities to share and obtain good quality infrastructure monitoring models without needing extensive capital and resources, and also allows communities to improve the robustness of their models by training them on diverse events

observed in other communities without the need to share sensitive data. FedGen addresses the associated challenges of centralized training, privacy concerns and data biases by leveraging federated learning as a distributed training paradigm to avoid data sharing, and uses a masking function driven by feature stability to identify and ignore biases in the training data. Our results indicate that FedGen results in significantly more generalizable monitoring models compared to other existing distributed and centralized training approaches.

Chapter 7

Conclusion

7.1 Summary of Thesis

In this thesis, we propose approaches towards building comprehensive infrastructure monitoring frameworks for smart communities. These frameworks involve a synergy between different layers of components, ranging from sensing to analysis, and achieving this goal requires optimizing these layers, while leveraging the unique structural and behavioral characteristics of different community infrastructure. Our efforts in this thesis, described below, are a key step towards building the next generation of infrastructure monitoring frameworks, that can be deployed anytime, anywhere, and for any application, while adapting to changing infrastructure conditions and resource availability.

We addressed the challenge of sensor placement using both in-situ and mobile sensors in Chapter 4, where we proposed several methodologies to measure the impact of adverse events on a given community, and thereby developed an impact-driven approach to determining the optimal locations for sensor deployment to rapidly detect high impact events on the community. We evaluated our approach on several real-world water distribution network

testbeds, and demonstrated that our approach resulted in a significantly lower impact of events on the community compared to traditional sensor placement approaches.

Chapter 5 presented REAM, our framework for resource efficient adaptive monitoring. It provides communities the ability to define custom monitoring pipelines consisting of sensors, network links, and analytical models at the edge or the cloud. REAM utilizes reinforcement learning agents to learn the various structural and behavioural patterns of the community infrastructure to make optimal decisions of which monitoring pipeline to execute for each of the multiple monitoring applications by looking at a quality vs. resource availability tradeoff. We deployed REAM on two real-world testbeds and demonstrate its ability to learn these patterns, and adapt to changing conditions. Our results showed that using REAM can enable communities to achieve high monitoring performance, while at the same time resulting in the judicious utilization of limited available resources.

In Chapter 6, we described the importance of developing approaches to share infrastructure monitoring models across communities. We explain challenges with centralized training, privacy requirements, and data biases that have thus far prevented an effective solution for the sharing or generalization of models across communities. We presented our approach, FedGen, that used a novel masking function to identify and ignore community-specific data biases in order to ensure that models would continue to perform well when deployed in other communities. FedGen also leveraged the federated learning paradigm, to overcome the drawbacks of centralized training and handle privacy concerns, wherein monitoring models could be trained in a distributed manner on data from infrastructure across multiple communities, without the need for sharing any raw data. We evaluated FedGen on several real-world monitoring use-cases and demonstrated its ability to train robust and generalizable models in a distributed manner.

7.2 Future Work

As we move towards a vision of developing better infrastructure monitoring solutions, that incorporate the different ideas proposed in this thesis, other challenges remain open. We outline several future directions that are quite interesting and important to pursue towards this vision.

7.2.1 Community Data Exchange and Interoperability

Today's infrastructures suffer from the issue of fragmented management, where they are maintained and operated by different entities (e.g., public agencies). Each of them develop solutions for their own infrastructure or location, and there is very little coordination. However, events in one infrastructure can often cascade into others. For instance, large scale flooding due to pipe breaks can cause damage to nearby industries and homes, and also affect supply chains of goods and services.

Developing data exchange solutions that allow seamless exchange of relevant information can help unify the management of different infrastructures and improve their efficiency and resilience when handling events. Achieving this requires solutions for interoperability of network and data exchange protocols. Since, each infrastructure and application can have their own data and information formats, facilitating exchanges would require translation from one to another. Rather than hard-coded approaches to implement translators for each pair of protocols, dynamically-configurable software artifacts can improve the flexibility and ease of data exchange.

7.2.2 Improved Planning and Maintenance

The vast majority of community infrastructure have been built without any thought for IoT instrumentation or monitoring. The use of sensor technology is now an afterthought where infrastructures are being retrofitted with devices, networks, and compute capabilities. This often leads to suboptimal results such as the inability to instrument certain locations or intermittent network connectivity due to poor planning. With the advantages of IoT-driven monitoring now becoming clear, it is important to improve the planning of new community infrastructure by incorporating the components of a monitoring workflow into the planning process from the beginning. Some initial efforts [31] have shown how infusing traditional urban planning tools with the cross-layer IoT workflow can result in improved monitoring performance while resulting in lower overall costs.

Additionally, the performance of different components of the monitoring workflow can degrade over time, necessitating solutions for periodic maintenance. For instance, the quality of measurements from sensors can deteriorate due to factors like mechanical wear or damage, environmental conditions like temperature and pressure, or even sudden electrical surges. Sensor calibration solutions are important to ensure the accuracy and validity of monitoring data which is essential to take appropriate decisions. Similarly, analytical models can also drift over time, This can happen due to concept drift, where the underlying physical phenomena of the event change, or data drift, where the statistical properties of the data measurements change over time. For example, climate change can cause a slow shift in the precipitation patterns of a community (concept drift), and can also cause an increase in overall temperature measurements of the region (data drift). Ensuring the effectiveness of an existing infrastructure monitoring framework requires addressing these planning and maintenance issues.

7.2.3 Cryptography for Secure Distributed Training

We addressed privacy challenges associated with infrastructure monitoring using the federated learning paradigm in Chapter 6. Federated learning only provides privacy to the extent that data is not shared across public networks with other entities. However, model parameters need to be transmitted, and these parameters become vulnerable to malicious actors who can introduce adversarial values that can prevent proper aggregation, or even worse, can induce the central model to emphasize wrong features entirely. This is an important issue especially when considering critical infrastructure like water networks, where incorrect model predictions can cause significant harm.

Cryptographic protocols that encrypt transmissions are a potential solution to help provide high security guarantees and further deter malicious actors. Techniques like Multi-Party Computation (MPC) provide methods for parties to jointly work together in a secure manner, while at the same time also protecting their private data from each other, which is a key tenet of this setting. Leveraging blockchain technology is another promising direction, where local model learning updates are periodically verified. This can ensure that in the event a local client is compromised, and adversarial samples are introduced to impact the central model, its parameters and updates can be verified by the other clients thereby detecting abnormal activity. Such techniques can overcome the single point of failure issue associated with today's distributed training approaches.

Bibliography

[1] Harvey caused a chemical plant explosion. is that the next face of climate change? https://www.washingtonpost.com/news/monkey-cage/wp/2017/09/06/harvey-caused-a-chemical-plant-explosion-is-that-the-next-face-of-\climate-change/. [Online; Accessed: 2021-10-08].

[2] K. Ahuja, K. Shanmugam, K. Varshney, and A. Dhurandhar. Invariant risk minimization games. In *International Conference on Machine Learning*, pages 145–155. PMLR, 2020.

[3] Airnow. Air quality index, 2021.

[4] I. Albuquerque, J. Monteiro, T. H. Falk, and I. Mitliagkas. Adversarial target-invariant representation learning for domain generalization. *arXiv preprint arXiv:1911.00804*, 2019.

[5] N. S. Alhassoun, M. Y. S. Uddin, and N. Venkatasubramanian. Safer: An iot-based perpetual safe community awareness and alerting network. In *2017 Eighth International Green and Sustainable Computing Conference (IGSC)*, pages 1–8. IEEE, 2017.

[6] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, et al. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

[7] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[8] A. W. W. Association. Buried no longer: confronting america's water infrastructure challenge. http://www.awwa.org/infrastructure. [Online; accessed 30-Oct-2013].

[9] B. Aydin, E. Selvi, J. Tao, and M. J. Starek. Use of fire-extinguishing balls for a conceptual system of drone-assisted wildfire fighting. *Drones*, 3(1):17, 2019.

[10] V. Bahl. Cloud 2020: Emergence of micro data centers (cloudlets) for latency sensitive computing. In *Middleware*, 2015.

[11] S. Beery, G. Van Horn, and P. Perona. Recognition in terra incognita. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 456–473, 2018.

[12] L. Begnudelli, B. F. Sanders, and S. F. Bradford. Adaptive godunov-based model for flood simulation. *Journal of Hydraulic Engineering*, 134(6):714–725, 2008.

[13] P. Bellavista, A. Corradi, L. Foschini, and A. Pernafini. Data distribution service (dds): A performance comparison of opensplice and rti implementations. In *2013 IEEE symposium on computers and communications (ISCC)*, pages 000377–000383. IEEE, 2013.

[14] K. Benson, C. Fracchia, G. Wang, Q. Zhu, S. Almomen, J. Cohn, L. D'arcy, D. Hoffman, M. Makai, J. Stamatakis, et al. Scale: Safe community awareness and alerting leveraging the internet of things. *IEEE Communications Magazine*, 53(12):27–34, 2015.

[15] K. E. Benson, G. Bouloukakis, C. Grant, V. Issarny, S. Mehrotra, I. Moscholios, and N. Venkatasubramanian. Firedex: A prioritized iot data exchange middleware for emergency response. In *Proceedings of the 19th International Middleware Conference*, pages 279–292, 2018.

[16] K. E. Benson, Q. Han, K. Kim, P. Nguyen, and N. Venkatasubramanian. Resilient overlays for iot-based community infrastructure communications. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 152–163. IEEE, 2016.

[17] K. E. Benson, G. Wang, N. Venkatasubramanian, and Y.-J. Kim. Ride: A resilient iot data exchange middleware leveraging sdn and edge cloud resources. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 72–83. IEEE, 2018.

[18] J. Berry, W. E. Hart, C. A. Phillips, J. G. Uber, and J.-P. Watson. Sensor placement in municipal water networks with temporal integer programming models. *Journal of water resources planning and management*, 132(4):218–224, 2006.

[19] J. W. Berry, L. Fleischer, W. E. Hart, C. A. Phillips, and J.-P. Watson. Sensor placement in municipal water networks. *Journal of Water Resources Planning and Management*, 131(3):237–243, 2005.

[20] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[21] A. Boretti and L. Rosa. Reassessing the projections of the world water development report. *NPJ Clean Water*, 2(1):1–6, 2019.

[22] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[23] T. Brinkhoff. City population. *Available online: www.city-population.de*, 2017.

[24] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

[25] L. J. Butler, M. K. Scammell, and E. B. Benson. The flint, michigan, water crisis: A case study in regulatory failure and environmental injustice. *Environmental Justice*, 9(4):93–97, 2016.

[26] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[27] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

[28] A. Cenedese, A. Zanella, L. Vangelista, and M. Zorzi. Padova smart city: An urban internet of things experimentation. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE, 2014.

[29] A. N. Center. Alabama power smart grid technology reduces customer outages following hurricane ida. `alabamanewscenter.com/2021/09/08/alabama-power-smart-grid-technology-reduces-customer-outages`. [Online; Accessed: 2021-10-08].

[30] S. Challenge. `https://www.smartamerica.org`. [Online; accessed 12-Oct-2021].

[31] T.-C. Chang, G. Bouloukakis, C.-Y. Hsieh, C.-H. Hsu, and N. Venkatasubramanian. Smartparcels: Cross-layer iot planning for smart communities. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, pages 195–207, 2021.

[32] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar. A dynamic service migration mechanism in edge cognitive computing. *ACM Transactions on Internet Technology (TOIT)*, 19(2):1–15, 2019.

[33] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.

[34] Y. J. Choe, J. Ham, and K. Park. An empirical study of invariant risk minimization. *arXiv preprint arXiv:2004.05007*, 2020.

[35] F. Chollet et al. Keras: The python deep learning library. *ascl*, pages ascl–1806, 2018.

[36] T. Chou, I. Ku, C. Wu, L.-C. Hsu, Y. Lin, Y. Chen, T.-Y. Huang, and C.-T. King. Canpas: a campus navigation and parking assistant system. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 631–638. IEEE, 2006.

[37] R. W. Clayton, T. Heaton, M. Kohler, M. Chandy, R. Guy, and J. Bunn. Community seismic network: A dense array to sense earthquake strong motion. *Seismological Research Letters*, 86(5):1354–1363, 2015.

[38] G. D'Angelo, S. Ferretti, and V. Ghini. Simulation of the internet of things. In *HPCS 2016*, pages 1–8, 2016.

[39] H. Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[40] S. S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, volume 3, pages 1609–1614. IEEE, 2003.

[41] W. Dong, G. Guan, Y. Chen, K. Guo, and Y. Gao. Mosaic: Towards city scale sensing with mobile sensor networks. In *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, pages 29–36. IEEE, 2015.

[42] Q. Dou, D. C. de Castro, K. Kamnitsas, and B. Glocker. Domain generalization via model-agnostic learning of semantic features. In *Advances in Neural Information Processing Systems*, pages 6450–6461, 2019.

[43] R. Du, C. Fischione, and M. Xiao. Flowing with the water: On optimal monitoring of water distribution networks by mobile sensors. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[44] R. Du, L. Gkatzikis, C. Fischione, and M. Xiao. Energy efficient sensor activation for water distribution networks based on compressive sensing. *IEEE Journal on Selected Areas in Communications*, 33(12):2997–3010, 2015.

[45] G. Dulac-Arnold, R. Evans, et al. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[46] Environmental Protection Agency. Open data us epa, 2021.

[47] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[48] R. Fletcher and M. Chandrasekaran. Smartball™: A new approach in pipeline leak detection. In *2008 7th International Pipeline Conference*, pages 117–133. American Society of Mechanical Engineers, 2008.

[49] K. Gai, M. Qiu, M. Liu, and H. Zhao. Smart resource allocation using reinforcement learning in content-centric cyber-physical systems. In *International Conference on Smart Computing and Communication*, pages 39–52. Springer, 2017.

[50] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[51] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[52] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[53] P. Gupta, S. Mehrotra, N. Panwar, S. Sharma, N. Venkatasubramanian, and G. Wang. Quest: Practical and oblivious mitigation strategies for covid-19 using wifi datasets. *arXiv preprint arXiv:2005.02510*, 2020.

[54] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. De Jesús. *Neural network design*, volume 20. Pws Pub. Boston, 1996.

[55] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[56] Q. Han et al. Aquaeis: Middleware support for event identification in community water infrastructures. In *Middleware*, 2019.

[57] Q. Han, P. Nguyen, R. T. Eguchi, K.-L. Hsu, and N. Venkatasubramanian. Toward an integrated approach to localizing failures in community water networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1250–1260. IEEE, 2017.

[58] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu. Health monitoring and management using internet-of-things (iot) sensing with cloud-based processing: Opportunities and challenges. In *2015 IEEE International Conference on Services Computing*, pages 285–292. IEEE, 2015.

[59] C. Heinze-Deml, J. Peters, and N. Meinshausen. Invariant causal prediction for nonlinear models. *Journal of Causal Inference*, 6(2), 2018.

[60] H. Hong, P. Tsai, et al. Supporting Internet-of-Things analytics in a fog computing platform. In *CloudCom'17*, 2017.

[61] M. S. Hossain and G. Muhammad. Cloud-assisted industrial internet of things (iiot)–enabled framework for health monitoring. *Computer Networks*, 101:192–202, 2016.

[62] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. *Mobile networks and Applications*, 10(4):519–528, 2005.

[63] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

[64] R.-T. Innovations. Rti connext dds. *Accessed: Apr*, 16:2018, 2018.

[65] R.-T. Innovations. `http://community.rti.com/rti-doc/510/ndds/doc/html/api_cpp/index.html`, 2021.

[66] K. Javed, M. White, and Y. Bengio. Learning causal models online. *arXiv preprint arXiv:2006.07461*, 2020.

[67] Kaggle. Air quality data, 2021.

[68] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[69] S. Kartakis, E. Abraham, and J. A. McCann. Waterbox: A testbed for monitoring and controlling smart water networks. In *Proceedings of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks*, page 8. ACM, 2015.

[70] A. Kessler, A. Ostfeld, and G. Sinai. Detecting accidental contaminations in municipal water networks. *Journal of Water Resources Planning and Management*, 124(4):192–198, 1998.

[71] N. A. Khan, N. Jhanjhi, S. N. Brohi, R. S. A. Usmani, and A. Nayyar. Smart traffic monitoring system using unmanned aerial vehicles (uavs). *Computer Communications*, 157:434–443, 2020.

[72] A. Khosla, T. Zhou, T. Malisiewicz, A. A. Efros, and A. Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012.

[73] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[74] M. D. Kohler, T. H. Heaton, and M.-H. Cheng. The community seismic network and quake-catcher network: Enabling structural health monitoring through instrumentation by community participants. In *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2013*, volume 8692, page 86923X. International Society for Optics and Photonics, 2013.

[75] B. Korte, J. Vygen, B. Korte, and J. Vygen. *Combinatorial optimization*, volume 2. Springer, 2012.

[76] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

[77] D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. Le Priol, and A. Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[78] A. Kumar, M. Kansal, G. Arora, A. Ostfeld, and A. Kessler. Detecting accidental contaminations in municipal water networks. *Journal of Water Resources Planning and Management*, 125(5):308–310, 1999.

[79] T. T.-T. Lai, W.-J. Chen, Y.-H. T. Chen, P. Huang, and H.-H. Chu. Mapping hidden water pipelines using a mobile sensor droplet. *ACM Transactions on Sensor Networks (TOSN)*, 9(2):1–33, 2013.

[80] A. Lambert. What do we know about pressure-leakage relationships in distribution systems. In *IWA Conf. n Systems approach to leakage control and water distribution system management*, 2001.

[81] K. L. Leuth. State of iot 2018: Number of iot devices now at 7b–accelerating. *https://iot-analytics. com/stateof-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/*, 2018.

[82] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 3490–3497. AAAI press, 2018.

[83] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[84] H. Li, S. Jialin Pan, S. Wang, and A. C. Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[85] P. W. Li, H. L. Zhao, H. T. Yang, and S. Sun. Performance evaluation of transport protocol in data distribution service middleware. In *Advanced Materials Research*, volume 926, pages 1984–1987. Trans Tech Publ, 2014.

[86] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[87] C.-C. Liao, T.-F. Hou, T.-Y. Lin, Y.-J. Cheng, A. Erbad, C.-H. Hsu, and N. Venkatasubramania. Sais: Smartphone augmented infrastructure sensing for public safety and sustainability in smart cities. In *Proceedings of the 1st International Workshop on Emerging Multimedia Applications and Services for Smart Cities*, pages 3–8, 2014.

[88] R. Liemberger, P. Marin, et al. The challenge of reducing non-revenue water in developing countries–how the private sector can help: A look at performance-based service contracting. 2006.

[89] S. Lin, G. Yang, and J. Zhang. A collaborative learning framework via federated meta-learning. In *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pages 289–299. IEEE, 2020.

[90] Z. C. Lipton, Y.-X. Wang, and A. Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018.

[91] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang. Privacy-preserving traffic flow prediction: A federated learning approach. *IEEE Internet of Things Journal*, 7(8):7751–7763, 2020.

[92] W. Luping, W. Wei, and L. Bo. Cmfl: Mitigating communication overhead for federated learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 954–964. IEEE, 2019.

[93] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56, 2016.

[94] E. Map. `http://elevationmap.net`.

[95] A. Marjovi, A. Arfire, and A. Martinoli. High resolution air pollution maps in urban environments using mobile sensor networks. In *2015 international conference on distributed computing in sensor systems*, pages 11–20. IEEE, 2015.

[96] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[97] S. Mehrotra, A. Kobsa, N. Venkatasubramanian, and S. R. Rajagopalan. Tippers: A privacy cognizant iot environment. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2016.

[98] D. Misiunas. *Failure monitoring and asset condition assessment in water supply systems*. Lund University, 2005.

[99] V. Mnih, K. Kavukcuoglu, et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[100] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR, 2019.

[101] I. Narayanan, A. Vasan, V. Sarangan, et al. One meter to find them all-water network leak localization using a single flow meter. In *Information Processing in Sensor Networks*, pages 47–58. IEEE, 2014.

[102] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[103] N. Nesa and I. Banerjee. Sensorrank: an energy efficient sensor activation algorithm for sensor data fusion in wireless networks. *IEEE Internet of Things Journal*, 6(2):2532–2539, 2018.

[104] T. Nishio and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[105] C. of Water Systems University of EXETER. `http://emps.exeter.ac.uk/engineering/research/cws/downloads/benchmarks/`.

[106] N. Oliker and A. Ostfeld. Inclusion of mobile sensors in water distribution system monitoring operations. *Journal of Water Resources Planning and Management*, 142(1):04015044, 2016.

[107] A. Ostfeld, J. G. Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J.-P. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan, et al. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 134(6):556–568, 2008.

[108] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[109] J. Pearl. *Causality:models, reasoning, and inference*. Cambridge university press, 2009.

[110] F. Pedregosa, G. Varoquaux, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2011.

[111] X. Peng, Z. Huang, Y. Zhu, and K. Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[112] L. Perelman and A. Ostfeld. Operation of remote mobile sensors for security of drinking water distribution systems. *Water research*, 47(13):4217–4226, 2013.

[113] L. S. Perelman, W. Abbas, X. Koutsoukos, and S. Amin. Sensor placement for fault location identification in water networks: A minimum test cover approach. *Automatica*, 72:166–176, 2016.

[114] R. Pérez, V. Puig, J. Pascual, A. Peralta, E. Landeros, and L. Jordanas. Pressure sensor distribution for leak detection in barcelona water distribution network. *Water science and technology: water supply*, 9(6):715–721, 2009.

[115] R. Pérez, V. Puig, J. Pascual, J. Quevedo, E. Landeros, and A. Peralta. Methodology for leakage isolation using pressure sensitivity analysis in water distribution networks. *Control Engineering Practice*, 19(10):1157–1167, 2011.

[116] G. Pettet, A. Mukhopadhyay, M. J. Kochenderfer, and A. Dubey. Hierarchical planning for resource allocation in emergency response systems. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pages 155–166, 2021.

[117] R. Pinzinger, J. Deuerlein, A. Wolters, and A. Simpson. Alternative approaches for solving the sensor placement problem in large networks. In *World Environmental and Water Resources Congress 2011: Bearing Knowledge for Sustainability*, pages 314–323, 2011.

[118] V. Piratla, P. Netrapalli, and S. Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. *arXiv preprint arXiv:2003.12815*, 2020.

[119] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll. Opc ua versus ros, dds, and mqtt: performance evaluation of industry 4.0 protocols. In *2019 IEEE International Conference on Industrial Technology (ICIT)*, pages 955–962. IEEE, 2019.

[120] M. M. Rahman, M. Y. Ahmed, T. Ahmed, B. Islam, V. Nathan, K. Vatanparvar, E. Nemati, D. McCaffrey, J. Kuang, and J. A. Gao. Breatheasy: Assessing respiratory diseases using mobile multimodal sensors. In *Proceedings of the 2020 International Conference on Multimodal Interaction*, pages 41–49, 2020.

[121] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 1421–1429. IEEE, 2018.

[122] A. Rasekh, R. Wu, W. W. A. W. Salim, and M. K. Banks. Operation of mobile sensors for monitoring municipal drinking water distribution systems. In *World Environmental and Water Resources Congress 2014*, pages 362–367, 2014.

[123] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *International Conference on Learning Representations*, 2017.

[124] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[125] L. Ricaurte. The array of things, chicago. *Urban Planning for Transitions*, pages 171–182, 2021.

[126] N. Rieke, J. Hancox, W. Li, F. Milletari, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, et al. The future of digital health with federated learning. *NPJ digital medicine*, 3(1):1–7, 2020.

[127] L. A. Rossman et al. Epanet 2: users manual. 2000.

[128] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 3:3, 2018.

[129] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications*, 68(2):1146–1159, 2019.

[130] B. Sanders and L. Begnudelli. Brezo: A hydrodynamic flood simulation algorithm, 2010.

[131] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[132] R. Sarrate, V. Puig, T. Escobet, and A. Rosich. Optimal sensor placement for model-based fault detection and isolation. In *2007 46th IEEE Conference on Decision and Control*, pages 2584–2589. IEEE, 2007.

[133] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[134] S. Shekhar, A. Chhokra, H. Sun, A. Gokhale, A. Dubey, X. Koutsoukos, and G. Karsai. Urmila: Dynamically trading-off fog and edge resources for performance and mobility-aware iot services. *Journal of Systems Architecture*, 107:101710, 2020.

[135] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*, pages 92–104. Springer, 2018.

[136] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Workshop on Applied Computational Geometry*, pages 203–222. Springer, 1996.

[137] S. P. Singh, T. Jaakkola, et al. Reinforcement learning with soft state aggregation. In *NIPS*, 1995.

[138] E. Siow, T. Tiropanis, and W. Hall. Analytics for the internet of things: A survey. *ACM Comput. Surv.*, 51(4):74:1–74:36, July 2018.

[139] M. Srivastava, T. Hashimoto, and P. Liang. Robustness to spurious correlations via human annotations. In *International Conference on Machine Learning*, pages 9109–9119. PMLR, 2020.

[140] I. Stoianov, L. Nachman, S. Madden, T. Tokmouline, and M. Csail. Pipenet: A wireless sensor network for pipeline monitoring. In *2007 6th International Symposium on Information Processing in Sensor Networks*, pages 264–273. IEEE, 2007.

[141] M. Sulc and J. Matas. Improving cnn classifiers by estimating test-time priors. In *Proceedings of ICCV Workshops*, pages 0–0, 2019.

[142] Y. Sun, H. Song, et al. Internet of things and big data analytics for smart and connected communities. *IEEE access*, 2016.

[143] M. A. Suresh, R. Stoleru, E. M. Zechman, and B. Shihada. On event detection and localization in acyclic flow networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(3):708–723, 2013.

[144] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[145] D. Teney, E. Abbasnejad, and A. v. d. Hengel. Unshuffling data for improved generalization. *arXiv preprint arXiv:2002.11894*, 2020.

[146] J. Thornton, R. Sturm, and G. Kunkel. *Water loss control*. McGraw-Hill Education, 2008.

[147] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[148] L. Times. Arizona power company baffled by events that led to outage. `https://latimesblogs.latimes.com/lanow/2011/09/blackout-san-diego-arizona.html`. [Online; Accessed: 2021-10-08].

[149] N. Y. Times. Cyberattacks put russian fingers on the switch at power plants, u.s. says. `https://www.nytimes.com/2018/03/15/us/politics/russia-cyberattacks.html`. [Online; accessed 12-Oct-2021].

[150] N. Y. Times. 'dangerous stuff': Hackers tried to poison water supply of florida town. `https://www.nytimes.com/2021/02/08/us/oldsmar-florida-water-supply-hack.html`. [Online; accessed 12-Oct-2021].

[151] A. Trivedi, C. Zakaria, R. Balan, A. Becker, G. Corey, and P. Shenoy. Wifitrace: Network-based contact tracing for infectious diseases using passive wifi sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1):1–26, 2021.

[152] M.-H. Tsai, N. Venkatasubramanian, and C.-H. Hsu. Analytics-aware storage of surveillance videos: Implementation and optimization. In *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 25–32. IEEE, 2020.

[153] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[154] B. Urbonas and P. Stahre. *Stormwater: best management practices and detention for water quality, drainage, and CSO management*. 1993.

[155] USEPA. Potential contamination due to cross-connections and backflow and the associated health risks, 2002.

[156] R. Vaisenberg, A. Della Motta, S. Mehrotra, and D. Ramanan. Scheduling sensors for monitoring sentient spaces using an approximate pomdp policy. *Pervasive and Mobile Computing*, 10:83–103, 2014.

[157] V. Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992.

[158] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha, A. Kapoor, M. Sudarshan, and S. Stratman. Farmbeats: An iot platform for data-driven agriculture. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 515–529, 2017.

[159] N. Venkatasubramanian, C. A. Davis, and R. T. Eguchi. Designing community-based intelligent systems for water infrastructure resilience. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Advances in Resilient and Intelligent Cities*, pages 62–65, 2020.

[160] P. Venkateswaran, K. E. Benson, C.-Y. Hsieh, C.-H. Hsu, S. Mehrotra, and N. Venkatasubramanian. Ream: A framework for resource efficient adaptive monitoring of community spaces. *Pervasive and Mobile Computing*, 76:101459, 2021.

[161] P. Venkateswaran, Q. Han, R. T. Eguchi, and N. Venkatasubramanian. Impact driven sensor placement for leak detection in community water networks. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 77–87. IEEE, 2018.

[162] P. Venkateswaran, C.-H. Hsu, S. Mehrotra, and N. Venkatasubramanian. Ream: Resource efficient adaptive monitoring of community spaces at the edge using reinforcement learning. In *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 17–24. IEEE, 2020.

[163] P. Venkateswaran, V. Muthusamy, V. Isahagian, and N. Venkatasubramanian. Environment agnostic invariant risk minimization for classification of sequential datasets. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1615–1624, 2021.

[164] P. Venkateswaran, V. Muthusamy, V. Isahagian, and N. Venkatasubramanian. Robust and generalizable predictive models for business processes. In *International Conference on Business Process Management*, pages 105–122. Springer, 2021.

[165] P. Venkateswaran, M. A. Suresh, and N. Venkatasubramanian. Augmenting in-situ with mobile sensing for adaptive monitoring of water distribution networks. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 151–162, 2019.

[166] P. Venkateswaran and N. Venkatasubramanian. Iot enabled data exchange for stormwater systems. In *ABSTRACTS OF PAPERS OF THE AMERICAN CHEMICAL SOCIETY*, volume 258. AMER CHEMICAL SOC 1155 16TH ST, NW, WASHINGTON, DC 20036 USA, 2019.

[167] R. Ventura, V. Mallet, and V. Issarny. Assimilation of mobile phone measurements for noise mapping of a neighborhood. *The journal of the acoustical society of America*, 144(3):1279–1292, 2018.

[168] R. Ventura, V. Mallet, V. Issarny, P.-G. Raverdy, and F. Rebhi. Estimation of urban noise with the assimilation of observations crowdsensed by the mobile application ambiciti. In *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 255, pages 5444–5451. Institute of Noise Control Engineering, 2017.

[169] P. Verma and S. K. Sood. Cloud-centric iot based disease diagnosis healthcare framework. *Journal of Parallel and Distributed Computing*, 116:27–38, 2018.

[170] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in neural information processing systems*, pages 5334–5344, 2018.

[171] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[172] D. Washburn, U. Sindhu, S. Balaouras, R. A. Dines, N. Hayes, and L. E. Nelson. Helping cios understand "smart city" initiatives. *Growth*, 17(2):1–17, 2009.

[173] C. Westerlund and M. Viklander. Particles and associated metals in road runoff during snowmelt and rainfall. *Science of the Total Environment*, 362(1-3):143–156, 2006.

[174] A. J. Whittle, L. Girod, A. Preis, M. Allen, H. B. Lim, M. Iqbal, S. Srirangarajan, C. Fu, K. J. Wong, and D. Goldsmith. Waterwise@ sg: A testbed for continuous monitoring of the water distribution system in singapore. In *Water Distribution Systems Analysis 2010*, pages 1362–1378. 2010.

[175] Z. Xu, Z. Yang, J. Xiong, J. Yang, and X. Chen. Elfish: Resource-aware federated learning on heterogeneous edge devices. *Ratio*, 2(r1):r2, 2019.

[176] J. Yang, H. Zou, S. Cao, Z. Chen, and L. Xie. Mobileda: Toward edge-domain adaptation. *IEEE Internet of Things Journal*, 7(8):6909–6918, 2020.

[177] I. Yaqoob, E. Ahmed, M. H. ur Rehman, A. I. A. Ahmed, M. A. Al-garadi, M. Imran, and M. Guizani. The rise of ransomware and emerging security challenges in the internet of things. *Computer Networks*, 129:444–458, 2017.

[178] M. Younis and K. Akkaya. Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6(4):621–655, 2008.

[179] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[180] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 377–392, 2017.

[181] M. Zhang, H. Marklund, A. Gupta, S. Levine, and C. Finn. Adaptive risk minimization: A meta-learning approach for tackling group shift. *arXiv preprint arXiv:2007.02931*, 2020.

[182] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen. Cautionary tales on air-quality improvement in beijing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170457, 2017.

[183] T. Zhang and O. Ardakanian. A domain adaptation technique for fine-grained occupancy estimation in commercial buildings. In *Proceedings of the International Conference on Internet of Things Design and Implementation*, pages 148–159, 2019.

[184] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster. User perceptions of smart home iot privacy. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–20, 2018.

[185] Q. Zhu. *Exploiting Mobile Plus In-Situ Deployments in Community IoT Systems*. University of California, Irvine, 2019.

[186] Q. Zhu, M. Y. S. Uddin, Z. Qin, and N. Venkatasubramanian. Upload planning for mobile data collection in smart community internet-of-things deployments. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2016.